1225

RD&E

CENTER

Technical Report

No. 13427

ROBOTIC REFUELER ARM KINEMATICS, DYNAMICS AND GLOBAL POSITION CONTROL

FEBRUARY 1989

By James Aardema
U.S. Army Tank-Automotive Command
ATTN: AMSTA-RYA
Warren, MI 48397-5000

U.S. ARMY TANK-AUTOMOTIVE COMMAND
RESEARCH, DEVELOPMENT & ENGINEERING CENTER
Warren, Michigan 48397-5000

NOTICES

This report is not to be construed as an official Department of the Army position.

Mention of any trade names or manufacturers in this report shall not be construed as an official endorsement or approval of such products or companies by the U.S. Government.

Destroy this report when it is no longer needed. Do not return it to the originator.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1a. REPORT SECURITY CLASSIFICATION Unclassified	1b. RESTRICTIVE MARKINGS None
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION / AVAILABILITY OF REPORT Approved for public release: Distribution is unlimited
2b. DECLASSIFICATION / DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) 13427	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION U.S. Army Tank-Automotive Command 6b. OFFICE SYMBOL (If applicable) AMSTA-RYA	7a. NAME OF MONITORING ORGANIZATION U.S. Army Tank-Automotive Command
6c. ADDRESS (City, State, and ZIP Code) Warren, Michigan 48397-5000	7b. ADDRESS (City, State, and ZIP Code) Warren, Michigan 48397-5000
8a. NAME OF FUNDING / SPONSORING ORGANIZATION 8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (City, State, and ZIP Code)	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT ACCESSION NO.

11. TITLE (Include Security Classification)
Robotic Refueler Arm Kinematics, Dynamics, and Global Position Control

12. PERSONAL AUTHOR(S)
Aardema, James A.

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Final	FROM 9/88 TO 12/88	1989, January	158

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	Robotics, Kinematics, Dynamics, Control Refueler-Demonstrator Vehicle Simulation

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

Currently, the U.S. Army is engaged in incorporating advanced robotics into military vehicle systems. One aspect of robotics used in military vehicles includes the use of a robotic arm to perform tasks a human operator is unable to perform or to perform a task which may endanger the operator.

One application for a robotic arm was developed by the U.S. Army Tank-Automotive Command (TACOM) who initiated a project to use a robotic refueler arm, guided either by a computer or by an operator with a joystick, to remotely refuel combat vehicles in the forward battle area at a high flow rate.

In this paper, the kinematics, inverse kinematics, dynamics, and a global position control strategy for the robotic refueler arm was analyzed. A computer simulation program was written to test the control strategy.

20. DISTRIBUTION / AVAILABILITY OF ABSTRACT ☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION Unclassifed
22a. NAME OF RESPONSIBLE INDIVIDUAL JAMES A. AARDEMA	22b. TELEPHONE (Include Area Code) (313) 574-5032 22c. OFFICE SYMBOL AMSTA-RYA

PREFACE

This analysis was conducted and this report was prepared in conjunction with the Robotic Refueler software programs being developed by Analytical and Physical Simulation Branch at the U.S. Army Tank-Automotive Command and to fulfill the requirements for a final class project in the class "Special Topics in Robotics (EGR 595)" taken during the fall 1988 semester at Oakland University located in Rochester Michigan The author's attendance in this class was supported by the U.S. Army Tank-Automotive Command. The class instructor was You-Liang Gu from the Department of Electrical and Systems Engineering.

The author wishes to acknowledge and thank Professor You-Liang Gu for the instruction and assistance he provided on this project and for the good grade recieved.

Since most of the work in preparing this report was conducted at home, the author would like to dedicate this report to his wife Donna, who was very patient and supportive during these studies.

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

LIST OF ILLUSTRATIONS

1.0. INTRODUCTION

Currently the U.S. Army is engaged in incorporating adavanced robotics into military vehicle systems. One aspect of robotics used in military vehicles includes the use of a robotic arm to perform tasks a human operator is unable to perform or to perform tasks which may endanger the operator.

One application for a robotic arm was developed by the U.S. Army Tank Automotive Command (TACOM) which initiated a project to determine the feasibility of remotely refueling combat vehicles in the forward battle area at a high flow rate while providing crew protection against ballistic and Nuclear, Biological, and Chemical (NBC) hazards. To accomplish these goals a robotic refueler arm, guided either by a computer or by an operator with a joystick, is being developed and integrated into a Refueler Demonstrator (RD) vehicle.

Another application for a robotic arm is being considered under the Heavy Forces Modernization Program. Under this program a Combat Mobility Vehicle (CMV) is being proposed to demonstrate the applicability and integration of advanced vehicle and robotic technologies. One of the advanced technologies to be incorporated into the CMV includes a multi-use robotic excavating arm. The robotic excavating arm will allow the CMV to defeat complex obstacle systems or contruct defensive positions with improved vehicle performance and crew survivability.

Both the robotic refueler arm and the multi-use robotic excavating arm require the integration of advanced robotic kinematics, dynamics, controls, and sensing with the vehicle and crew.

Although this paper uses the RD vehicle for the analysis, the technology presented can be applied to the multi-use robotic excavating arm or any robotic arm.

This analysis was conducted and this report was prepared in conjunction with the Robotic Refueler software programs being developed by Analytical and Physical Simulation Branch at the U.S. Army Tank-Automotive Command and to fulfill the requirements for a final class project in the class "Special Topics in Robotics (EGR 595)" taken during the fall 1988 semester at Oakland University located in Rochester Michigan. The author's attendance in this class was supported by the U.S. Army Tank-Automotive Command. The class instructor was You-Liang Gu from the Department of Electrical and Systems Engineering.

2.0. OBJECTIVES

The objective of this analysis was to apply the theory and procedures developed in the above class to the robotic arm on the RD vehicle. For the robotic refueler arm, this study analyzed and developed a computer simulation program to calculate the following:

- Kinematics
- Inverse Kinematics
- Dynamics
- Global Position Control Stategy

3.0. CONCLUSIONS

The robotic refueler has 4 degrees of freedom. The four degrees of freedom result from the four joint angles and can be represented by the arm's ability to control the nozzle end position in the global X, Y, and Z directions and the nozzle approach angle. The nozzle approach angle is defined as the angle from horizontal of the nozzle in the robotic arm's plane of motion. Since the arm has only 4 degrees of freedom, the nozzle cannot be orientated outside the robotic arm's plane of motion. As a result, the nozzle can not be aligned with all receiver orientations.

To improve the robotic refueler arm's ability to align the nozzle with all receiver orientations, another degree of freedom must by built into the robotic refueler arm. This can be accomplished by adding another revolute joint between the elbow joint and the wrist joint. However, if only small angles outside the robotic arm's plane of motion are expected, then a compliant device connected between the nozzle and the wrist joint could be used.

A computer program was written in the "C" language to simulate the robotic refueler arm's kinematic, dynamic, and control equations and to evaluate the performance of the robot control system against disturbances.

A desired trajectory, which the controlled robot is to follow, was chosen to be a 72-inch radius circular path in the horizontal plane with a simultaneous 48-inch sinusoidal rise in elevation while maintaining a constant approach angle of 90 degrees. The initial conditions in the global X, Y, and Z direction were purposely offset from the desired trajectory by 12.0 inches and the nozzle orientation was

offset by 12 degrees to evaluate the performance of the control system against disturbances.

The simulation results show that the position error in the global X, Y, and Z direction and the orientation error goes to zero and the nozzle end follows the desired tractory.

4.0. RECOMMENDATIONS

In the future, further analysis should be performed on:

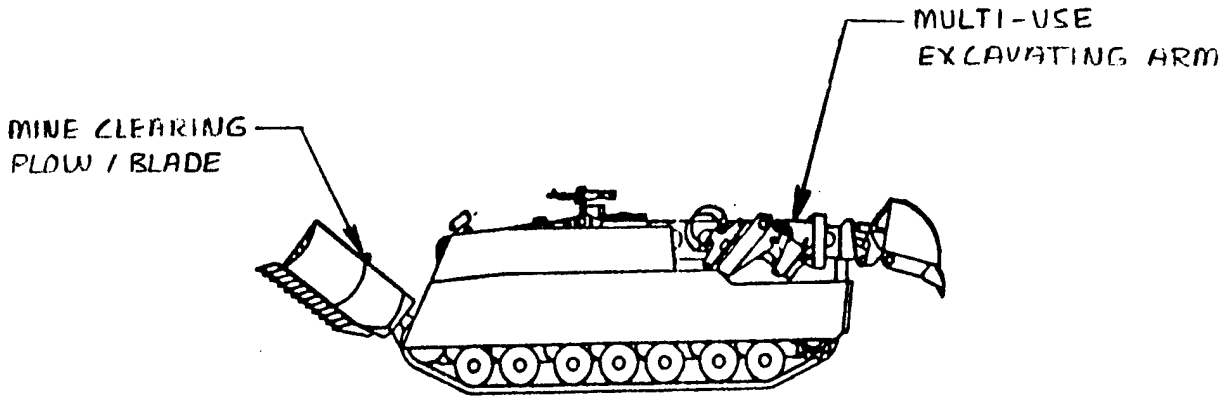
- System hydrualics.
- Incorporation of system hydraulics into a global positional control system.
- Positional controller for a single hydraulically driven joint.
- Trajectory planning and obstacle avoidance
- Camera model to characterize the formation of an image via the projection of 3D points onto an image plane.
- Integration of camera and other sensors with controller and with the vehicle and crew.

5.0. DISCUSSION

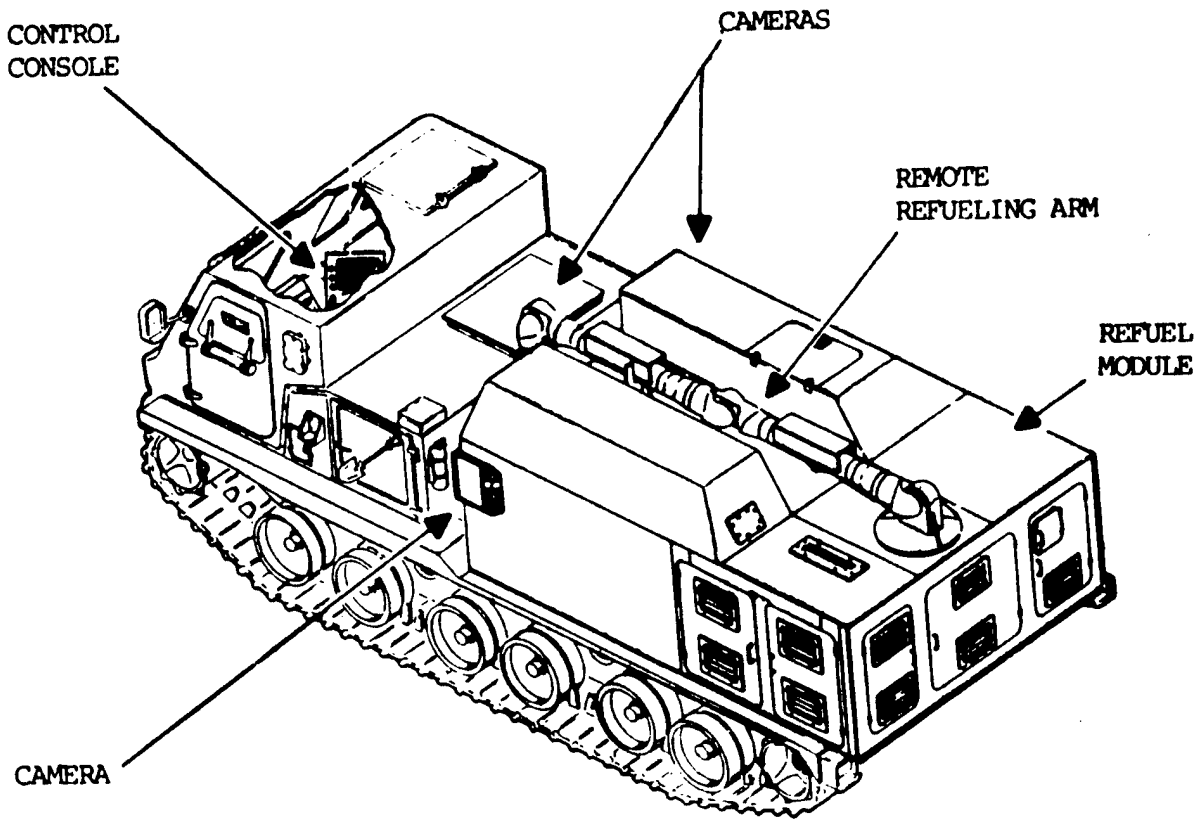
5.1. Combat Mobility Vehicle

5.1.1. Background. Under the Heavy Forces Modernization Program, a Combat Mobility Vehicle (CMV) is being proposed to demonstrate the applicability and integration of advanced vehicle and robotic technologies. Advanced technologies incorporated into the CMV include: multi-use robotic excavating arm, mine clearing plow/blade, hydraulic technology, advanced integration propulsion system, VETRONICS (vehicle electronics), advanced track and suspension system, and countermeasure technologies. These technologies will allow the CMV to defeat complex obstacle systems with improved vehicle and crew survivability, and with improved mobility comparable to other combat vehicles. Figure 5-1 shows a concept drawing of the CMV.

The multi-use robotic excavating arm requires the integration of advanced robotic kinematics, dynamics, controls, and sensing with the vehicle and crew. Although



(a) Combat Mobiltiy Vehicle (CMV)



(b) Refueler Demonstrator (RD)

Figure 5-1. Combat Mobility Vehicle and Refueler Demonstrator

this paper uses the Refueler Demonstrator vehicle for this analysis, the technology presented can be applied to any robotic arm.

5.2. Refueler Demonstrator Vehicle

5.2.1. Background. "During the development of the Armored Resupply Multipurpose System (ARMS) program, a major requirement evolved for forward area refueling. The ARMS solution was to use a conventional delivery system placed on an armored tracked carrier to provide increased mobility and armor protection. Looking beyond ARMS, the U.S. Army Tank Automotive Command (TACOM) developed a project to determine the feasibility of remotely refueling combat vehicles in the forward battle area. To achieve this, a Refueler Demonstrator (RD) was developed for engineering test and user evaluation. The RD project is being fully coordinated with the Belvoir Research, Development, and Engineering Center (BRDEC). The RD is also being used to evaluate the Standard Army Refueling System (SARS). Under this program, a number of interfaces such as nozzles, receptacles, and fuel flow rates will be evaluated. Currently, the RD is viewed as a technology brassboard and early-on proof of principle for the refueler variant of the Future Armored Resupply Vehicle (FARV) in the armored family of vehicles (AFV) (under the Heavy Forces Modernization Program)."[1]

5.2.2. Vehicle Description.

5.2.2.1. System Description. "The RD consists of a refueler module mounted on an M993 track vehicle chassis. (The RD is designed to refuel combat vehicles at a high flow rate and provide crew protection against ballistic and Nuclear Biological and Chemical (NBC) hazards). The system is shown in Figure 5-1. The fuel delivery system consists of the pumping components for automatic and manual remote refueling as well as a standard refueling system that incorporates a hose and reel. The automatic and manual remote system has a variable (fuel delivery) rate adjustment from 1 to 250 gallons per minute (GPM). The standard refueling system has an adjustable (fuel delivery) rate from 1 to 50 GPM. (In addition to refueling, the refueler is capable of defueling at a rate of approximately 50 GPM.) For remote refueling, the flow rate, fuel quantity, location of the vehicle to be refueled in relation to the RD (left or right) is entered into the computer by pressing a series of CRT prompt buttons. The operator is then asked whether the remote automatic or remote manual mode of operation is to be used. When the remote automatic mode of operation is selected, the refueling arm will automatically search for and locate the fuel receptacle on the vehicle to be

refilled. Location of the receptacle is accomplished by the refueling arm nozzle sensing infrared light beams which come from a device attached to the fuel receptacle. The angle of the nozzle is checked and aligned with the receiving vehicle. Text from the computer will then ask if the operator wants to guide the refueling arm into the receptacle manually, with a joy stick, or if the operator wants the remote refueling arm to automatically engage the receptacle. After pressing the proper CRT prompt button, the fuel receptacle is engaged. When the remote manual mode operation is selected, the refueling arm search and engagement process is totally controlled by means of a joystick."[2]

5.2.2.2. Refueler Module. "The refueler module is armor protected and houses a 1,500 gallon fuel tank, fuel delivery system, and self-contained auxiliary power unit (APU). Armored access doors are incorporated in both the pumping and APU compartments to provide access for service and maintenance. The fuel tank is integral with the armored module structure and includes and explosion suppressing material which also acts as a baffle."[3]

5.2.2.3. Cameras. "The RD contains three cameras. A camera is mounted on each side of the (refueler) module as part of the observation system to aid the operator in locating the fuel receptacle of the receiving vehicle during automatic and manual remote operations. These cameras are fixed facing rearward and have a wide field of view. A single narrow field of view camera is also mounted on the remote refueling arm to provide the operator with close in viewing for nozzle positioning."[4]

5.2.2.4. Remote Refueling Arm. "The remote refueling arm is mounted on the refueler module and provides for fuel delivery from the module to the receiving vehicle. It is constructed of armor tubing and can be automatically or manually operated throughout its deployment and receptacle engagement. The arm is hydraulically powered and controlled by electro-hydraulic actuators. Signals to the actuators are transmitted from the control console at the operator's control station. Transmitters/receivers are also mounted on the arm to provide the operator the necessary signals to locate the fuel receptacle and engage the refueling arm nozzle."[5]

5.2.2.5. Control Console. "The control console is in the vehicle cab between the driver and passenger seats. The control console consists of a computer, control cabinet containing a CRT monitor, joystick, and associated controls for performing remote refueling functions."[6]

5.3. Kinematics

5.3.1. Links, Joints, and Their Parameters. A mechanical manipulator consists of a sequence of rigid bodies, called links, connected by either revolute or prismatic joints. Each joint-link pair constitutes 1 degree of freedom. Hence, for the robotic refueler arm on the RD vehicle, there are 4 degrees of freedom as a result of the 4 revolute joints between the 4 bodies. Figure 5-2 shows a drawing of the refueler arm. The 4 links are: the rotating base; the aft arm; the fore arm; and the nozzle. The 4 joints are: the waist joint between the vehicle chassis and the rotating base; the shoulder joint between the rotating base and the aft arm; the elbow joint between the aft arm and the fore arm; the wrist joint between the fore arm and the nozzle.

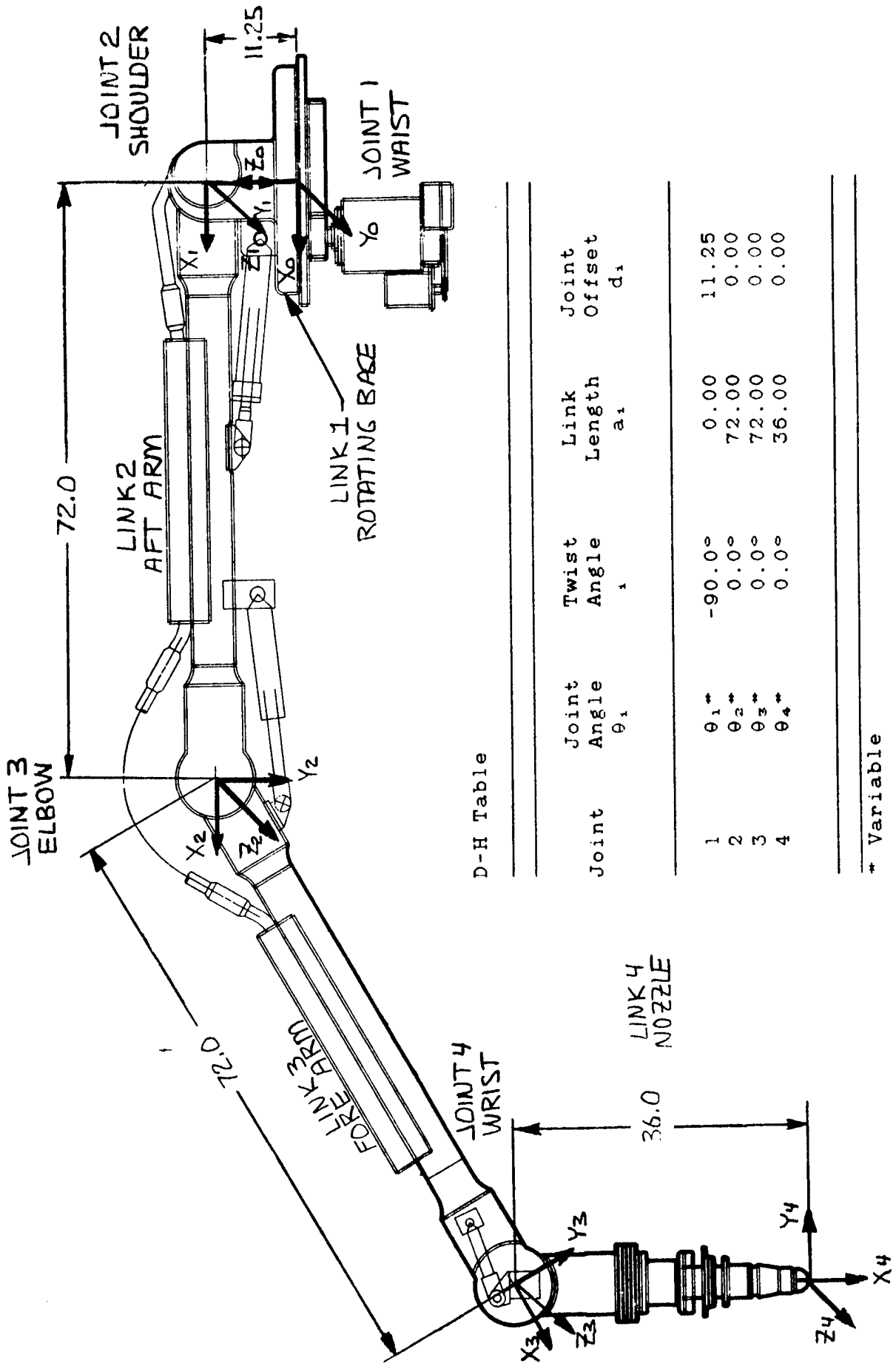
There are two parameters for each joint, Joint Offset (d_i) and Joint Angle (θ_i), which determines the relative position of neighboring links. Joint Offset is the relative position of 2 connected links (link i-1 and link i) which is the distance along the joint axis between normals. Joint Angle is the angle between the normals measured in a plane normal to the joint axis.

There are two parameters for each link, Link Length (a_i) and Twist Angle (α_i), which determines the structure of the link. Link Length is the shortest distance measured along the common normal between joint axes. Twist Angle is the angle between the joint axes in a plane perpendicular to the Link Length.

For a more complete description on each parameter the reader should refer to chapter 2 of the book "Robotics: Control, Sensing, Vision, and Intelligence" by Fu, Gonzalez, and Lee.

5.3.2. Denavit-Hartenberg (D-H) Representation. "To describe the translational and rotational relationships between adjacent links, Denavit and Hartenberg proposed a matrix method of systematically establishing a coordinate system (body attached frame) to each link of an articulated chain. The Denavit-Hartenberg representation results in a 4 by 4 homogenous transformation matrix representing each link's coordinate system at the joint with respect to the previous link's coordinate system. Thus, through sequential transformations, the end-effector (nozzle) expressed in nozzle coordinates can be transformed and expressed in the global (inertial, base, or vehicle) coordinates which make up the inertial frame of the dynamic system."[7]

"The homogenous transformation matrix is a 4 by 4 matrix which maps a position vector expressed in homogenous



D-H Table

Joint	Joint Angle θ_i	Twist Angle α_i	Link Length a_i	Joint Offset d_i
1	θ_1*	-90.0°	0.00	11.25
2	θ_2*	0.0°	72.00	0.00
3	θ_3*	0.0°	72.00	0.00
4	θ_4*	0.0°	36.00	0.00

* Variable

Figure 5-2. Link, Joints, and their Parameters

coordinates from one coordinate system to another coordinate system. A homogenous transformation matrix can be considered to consist of four submatrices:

$$\mathbf{A} = \begin{bmatrix} \mathbf{R}_{3x3} & \mathbf{P}_{3x1} \\ \mathbf{f}_{1x3} & 1x1 \end{bmatrix}$$

$$\mathbf{A} = \begin{bmatrix} n_x & s_x & a_x & p_x \\ n_y & s_y & a_y & p_y \\ n_z & s_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \mathbf{n} & \mathbf{s} & \mathbf{a} & \mathbf{p} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{A} = \begin{bmatrix} \text{rotation matrix} & \text{position vector} \\ \text{perspective transformation} & \text{scaling} \end{bmatrix}$$

The upper left 3 by 3 submatrix represents the rotation matrix; the upper right 3 by 1 submatrix represents the position vector of the origin of the rotated coordinate system with respect to the reference system; the lower left 1 by 3 submatrix represents perspective transformation; and the fourth diagonal element is the global scaling factor."[8]

"Since the inverse of an orthonormal rotation submatrix is equivalent to its transpose, the row vectors of a rotation submatrix represent the principal axes of the reference coordinate system with respect to the rotated coordinate system. However, the inverse of a non-orthogonal homogeneous transformation matrix is not equivalent to its transpose. The position of the origin of the reference coordinate system with respect to the rotated coordinate system can only be found after the inverse of the homogeneous transformation matrix is determined. In general the inverse transformation matrix can be found to be:

$$\mathbf{A}^{-1} = \begin{bmatrix} n_x & n_y & n_z & -\mathbf{n}^T\mathbf{p} \\ s_x & s_y & s_z & -\mathbf{s}^T\mathbf{p} \\ a_x & a_y & a_z & -\mathbf{a}^T\mathbf{p} \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} & \mathbf{R}_{3x3}^T & & -\mathbf{n}^T\mathbf{p} \\ & & & -\mathbf{s}^T\mathbf{p} \\ & & & -\mathbf{a}^T\mathbf{p} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The column vectors of the inverse of a homogenous transformation matrix represent the principal axes of the reference system with respect to the rotated coordinate system, and the upper right 3 by 1 submatrix represents the position of the orgin of the reference frame with respect to the rotated system."[9]

The D-H reprsentation of a rigid link is based upon a set of rules and upon the four geometric paramters mentioned previously. The reader should refer to the book "Robotics: Control, Sensing, Vision, and Intelligence" by Fu, Gonzalez, and Lee for the complete set of rules for establishing each link's coordinate system. The D-H reprsentation for the robotic refueler arm is given in Figure 5-2.

For a revolute joint, the homogenous transformation matrix relating the ith coordinate frame to the (i-1)th coordinate frame where $\emptyset_i, \alpha_i$, a_i, and d_i are the link and joint parameters of the ith system and $\emptyset_i$ is a variable is given by:

$$ {}^{i-1}\mathbf{A}_i = \begin{bmatrix} \cos\emptyset_i & -\cos\alpha_i*\sin\emptyset_i & \sin\alpha_i*\sin\emptyset_i & a_i*\cos\emptyset_i \\ \sin\emptyset_i & \cos\alpha_i*\cos\emptyset_i & -\sin\alpha_i*\cos\emptyset_i & a_i*\sin\emptyset_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

For a prismatic (translational) joint, the homogenous transformation matrix relating the ith coordinate frame to the (i-1)th coordinate frame where $\emptyset_i$, α_i, a_i, and d_i are the link and joint parameters of the ith system and d_i is a variable is given by:

$$ {}^{i-1}\mathbf{A}_i = \begin{bmatrix} \cos\emptyset_i & -\cos\alpha_i*\sin\emptyset_i & \sin\alpha_i*\sin\emptyset_i & 0 \\ \sin\emptyset_i & \cos\alpha_i*\cos\emptyset_i & -\sin\alpha_i*\cos\emptyset_i & 0 \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

For the robotic refueler arm shown in Figure 5-2, the homogenous transformation matrix relating coordinate system 1 to coordinate system 0 is given by:

$$
{}^{0}\mathbf{A}_{1} = \begin{bmatrix} \cos\emptyset_1 & 0 & -\sin\emptyset_1 & 0 \\ \sin\emptyset_1 & 0 & \cos\emptyset_1 & 0 \\ 0 & -1 & 0 & 11.25 \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

The homogenous transformation matrix relating coordinate system 2 to coordinate system 1 is given by:

$$
{}^{1}\mathbf{A}_{2} = \begin{bmatrix} \cos\emptyset_2 & -\sin\emptyset_2 & 0 & 72*\cos\emptyset_2 \\ \sin\emptyset_2 & \cos\emptyset_2 & 0 & 72*\sin\emptyset_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

The homogenous transformation matrix relating coordinate system 3 to coordinate system 2 is given by:

$$
{}^{2}\mathbf{A}_{3} = \begin{bmatrix} \cos\emptyset_3 & -\sin\emptyset_3 & 0 & 72*\cos\emptyset_3 \\ \sin\emptyset_3 & \cos\emptyset_3 & 0 & 72*\sin\emptyset_3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

The homogenous transformation matrix relating coordinate system 4 to coordinate system 3 is given by:

$$
{}^{3}\mathbf{A}_{4} = \begin{bmatrix} \cos\emptyset_4 & -\sin\emptyset_4 & 0 & 36*\cos\emptyset_4 \\ \sin\emptyset_4 & \cos\emptyset_4 & 0 & 36*\sin\emptyset_4 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

5.3.3. Position. The kinematic position analysis calculates, in the Cartesian space, the position of the

nozzle (coordinate system 4) with respect to the inertial system ($X_{4/0}$, $Y_{4/0}$, $Z_{4/0}$) given the joint angles $\emptyset_1$, $\emptyset_2$, $\emptyset_3$, and $\emptyset_4$.

The homogenous matrix ${}^0\mathbf{T}_i$, which specifies the location of the ith coordinate system with respect to the inertial coordinate system located on the vehicle, is the chain product of successive coordinate transformation matrices ${}^{i-1}\mathbf{A}^i$, and is expressed as:

$${}^0\mathbf{T}_i = {}^0\mathbf{A}_1 \; {}^1\mathbf{A}_2 \; \ldots \; {}^{i-1}\mathbf{A}_i$$

for i = 1,2,...,n where n is the number of links.

The homogenous transformation matrix relating coordinate system 2 to coordinate system 0 is given by:

$${}^0\mathbf{T}_2 = \begin{bmatrix} c1c2 & -c1s2 & -s1 & 72*c1c2 \\ s1c2 & -s1s2 & c1 & 72*s1c2 \\ -s2 & -c2 & 0 & -72*s2+11.25 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

where c1 = Cos($\emptyset_1$); c2 = Cos($\emptyset_2$);
s1 = Sin($\emptyset_1$); s2 = Sin($\emptyset_2$).

The homogenous transformation matrix relating coordinate system 3 to coordinate system 0 is given by:

$${}^0\mathbf{T}_3 = \begin{bmatrix} c1c2c3-c1s2s3 & -c1c2s3-c1s2c3 & -s1 & 72*(c1c2c3-c1s2s3+c1c2) \\ s1c2c3-s1s2s3 & -s1c2s3+s1s2c3 & c1 & 72*(s1c2c3-s1s2s3+s1c2) \\ -s2c3-c2s3 & s2s3-c2c3 & 0 & -72*(s2c3-c2s3-s2)+11.25 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

where c3 = Cos($\emptyset_3$); s3 = Sin($\emptyset_3$).

This can be simplified to:

$$
{}^{0}\mathbf{T}_{3} = \begin{bmatrix} c1c23 & -c1s23 & -s1 & 72*c1c23+72*c1c2) \\ s1c23 & -s1s23 & c1 & 72*s1c23+72*s1c2) \\ -s23 & -c23 & 0 & -72*s23-72*s2+11.25 \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

where:
$c23 = \cos(\emptyset_2+\emptyset_3) = \cos(\emptyset_2)\cos(\emptyset_3)-\sin(\emptyset_2)\sin(\emptyset_3)$
$s23 = \sin(\emptyset_2+\emptyset_3) = \sin(\emptyset_2)\cos(\emptyset_3)+\cos(\emptyset_2)\sin(\emptyset_3)$

The homogenous transformation matrix relating coordinate system 4 to coordinate system 0 is given by:

$$
{}^{0}\mathbf{T}_{4} = \begin{bmatrix} c1c234 & -c1s234 & -s1 & (36*c1c234 + 72*c1c23 + 72*c1c2) \\ s1c234 & -s1s234 & c1 & (36*s1c234 + 72*s1c23 + 72*s1c2) \\ -s234 & -c234 & 0 & (-36s234 - 72*s23 - 72*s2 + 11.25) \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

where: $c234 = \cos(\emptyset_2+\emptyset_3+\emptyset_4)$ and $s234 = \sin(\emptyset_2+\emptyset_3+\emptyset_4)$

The position of the nozzle (coordinate system 4) with respect to the inertial system ($x_{4/0}$, $y_{4/0}$, $z_{4/0}$), given the joint angles $\emptyset_1$, $\emptyset_2$, $\emptyset_3$, and $\emptyset_4$, is found by extracting the position vector from the transformation matrix ${}^{0}\mathbf{T}_{4}$. The approach angle ($\emptyset_a$) of the nozzle with respect to the receiver, which is more clearly defined in the Inverse Kinematics section, is calculated by: $\emptyset_a = \emptyset_2 + \emptyset_3 + \emptyset_4$. The position and orientation of the nozzle in Cartesian space is a function of the joint angles and can be written as:

$$\mathbf{y} = \mathbf{h}(\mathbf{q})$$

where $\mathbf{y}$ is the position vector in Cartesian Space

$$
\mathbf{y} = \begin{bmatrix} x_{4/0} \\ y_{4/0} \\ z_{4/0} \\ \emptyset_a \end{bmatrix}
$$

and where **q** is the joint position vector of variables in Joint Space

$$\mathbf{q} = \begin{bmatrix} \phi_1 \\ \phi_2 \\ \phi_3 \\ \phi_4 \end{bmatrix}$$

The position of the nozzle (coordinate system 4) in cartesian coordinates given the joint angles is equal to:

$$\mathbf{y} = \begin{bmatrix} x_{4/0} \\ y_{4/0} \\ z_{4/0} \\ \phi_a \end{bmatrix} = \begin{bmatrix} 36*c1c234 + 72*c1c23 + 72*c1c2 \\ 36*s1c234 + 72*s1c23 + 72*s1c2 \\ -36s234 - 72*s23 - 72*s2 + 11.25 \\ \phi_2 + \phi_3 + \phi_4 \end{bmatrix} = \begin{bmatrix} h_1 \\ h_2 \\ h_3 \\ h_4 \end{bmatrix}$$

5.3.4. Velocity. The kinematic velocity analysis calculates the velocity of the nozzle (coordinate system 4) with respect to the inertial system given the joint angles and joint velocities using:

$$\dot{\mathbf{y}} = (\partial \mathbf{h}/\partial \mathbf{q})\ \dot{\mathbf{q}} = \mathbf{Jh}\ \dot{\mathbf{q}}$$

where **Jh** is the Jacobian matrix defined by:

$$\mathbf{Jh} = \begin{bmatrix} \frac{\partial h_1}{\partial q_1} & \frac{\partial h_1}{\partial q_2} & \frac{\partial h_1}{\partial q_3} & \frac{\partial h_1}{\partial q_4} \\ \frac{\partial h_2}{\partial q_1} & \frac{\partial h_2}{\partial q_2} & \frac{\partial h_2}{\partial q_3} & \frac{\partial h_2}{\partial q_4} \\ \frac{\partial h_3}{\partial q_1} & \frac{\partial h_3}{\partial q_2} & \frac{\partial h_3}{\partial q_3} & \frac{\partial h_3}{\partial q_4} \\ \frac{\partial h_4}{\partial q_1} & \frac{\partial h_4}{\partial q_2} & \frac{\partial h_4}{\partial q_3} & \frac{\partial h_4}{\partial q_4} \end{bmatrix}$$

For the robotic refueler arm, the first row of the Jacobian matrix is:

Jh(1,1) = - 36s1c234 - 72s1c23 - 72s1c2

Jh(1,2) = - 36c1s234 - 72c1s23 - 72c1s2

Jh(1,3) = - 36c1s234 - 72c1s23

Jh(1,4) = - 36c1s234

The second row of the Jacobian matrix is:

Jh(2,1) = 36c1c234 + 72c1c23 + 72c1c2

Jh(2,2) = - 36s1s234 - 72s1s23 - 72s1s2

Jh(2,3) = - 36s1s234 - 72s1s23

Jh(2,4) = - 36s1s234

The third row of the Jacobian matrix is:

Jh(3,1) = 0

Jh(3,2) = - 36c234 - 72c23 - 72c2

Jh(3,3) = - 36c234 - 72c23

Jh(3,4) = - 36c234

The fourth row of the Jacobian matrix is:

Jh(4,1) = 0

Jh(4,2) = 1

Jh(4,3) = 1

Jh(4,4) = 1

5.3.5. Acceleration. The kinematic acceleration analysis calculates the acceleration of the nozzle (coordinate system 4) with respect to the inertial system given the joint angles, joint velocities, joint accelerations, and the time derivative of the Jacobian matrix using:

$$\ddot{\mathbf{y}} = \dot{\mathbf{Jh}}\,\dot{\mathbf{q}} + \mathbf{Jh}\,\ddot{\mathbf{q}}$$

The time derivative of the Jacobian matrix, denoted by **Jhd**, for the robotic refueler arm is given below:

The first row:

$$\begin{aligned}\mathbf{Jhd}(1,1) = &- 36c1c234 * \dot{\phi}_1 + 36s1s234 * (\dot{\phi}_2+\dot{\phi}_3+\dot{\phi}_4)\\ &- 72c1c23 * \dot{\phi}_1 + 72s1s23 * (\dot{\phi}_2+\dot{\phi}_3)\\ &- 72c1c2 * \dot{\phi}_1 + 72s1s2 * (\dot{\phi}_2)\end{aligned}$$

$$\begin{aligned}\mathbf{Jhd}(1,2) = &+ 36s1s234 * \dot{\phi}_1 - 36c1c234 * (\dot{\phi}_2+\dot{\phi}_3+\dot{\phi}_4)\\ &+ 72s1s23 * \dot{\phi}_1 - 72c1c23 * (\dot{\phi}_2+\dot{\phi}_3)\\ &+ 72s1s2 * \dot{\phi}_1 - 72c1c2 * (\dot{\phi}_2)\end{aligned}$$

$$\mathbf{Jhd}(1,3) = +\ 36s1s234 * \dot{\emptyset}_1 - 36c1c234 * (\dot{\emptyset}_2+\dot{\emptyset}_3+\dot{\emptyset}_4) + 72s1s23 * \dot{\emptyset}_1 - 72c1c23 * (\dot{\emptyset}_2+\dot{\emptyset}_3)$$

$$\mathbf{Jhd}(1,4) = +\ 36s1s234 * \dot{\emptyset}_1 - 36c1c234 * (\dot{\emptyset}_2+\dot{\emptyset}_3+\dot{\emptyset}_4)$$

The second row:

$$\mathbf{Jhd}(2,1) = -\ 36s1c234 * \dot{\emptyset}_1 - 36c1s234 * (\dot{\emptyset}_2+\dot{\emptyset}_3+\dot{\emptyset}_4) - 72s1c23 * \dot{\emptyset}_1 - 72c1s23 * (\dot{\emptyset}_2+\dot{\emptyset}_3) - 72s1c2 * \dot{\emptyset}_1 - 72c1s2 * (\dot{\emptyset}_2)$$

$$\mathbf{Jhd}(2,2) = -\ 36c1s234 * \dot{\emptyset}_1 - 36s1c234 * (\dot{\emptyset}_2+\dot{\emptyset}_3+\dot{\emptyset}_4) - 72c1s23 * \dot{\emptyset}_1 - 72s1c23 * (\dot{\emptyset}_2+\dot{\emptyset}_3) - 72c1s2 * \dot{\emptyset}_1 - 72s1c2 * (\dot{\emptyset}_2)$$

$$\mathbf{Jhd}(2,3) = -\ 36c1s234 * \dot{\emptyset}_1 - 36s1c234 * (\dot{\emptyset}_2+\dot{\emptyset}_3+\dot{\emptyset}_4) - 72c1s23 * \dot{\emptyset}_1 - 72s1c23 * (\dot{\emptyset}_2+\dot{\emptyset}_3)$$

$$\mathbf{Jhd}(2,4) = -\ 36c1s234 * \dot{\emptyset}_1 - 36s1c234 * (\dot{\emptyset}_2+\dot{\emptyset}_3+\dot{\emptyset}_4)$$

The third row:

$$\mathbf{Jhd}(3,1) = 0$$

$$\mathbf{Jhd}(3,2) = 36s234 * (\dot{\emptyset}_2+\dot{\emptyset}_3+\dot{\emptyset}_4) + 72s23 * (\dot{\emptyset}_2+\dot{\emptyset}_3) + 72s2 * (\dot{\emptyset}_2)$$

$$\mathbf{Jhd}(3,3) = 36s234 * (\dot{\emptyset}_2+\dot{\emptyset}_3+\dot{\emptyset}_4) + 72s23 * (\dot{\emptyset}_2+\dot{\emptyset}_3)$$

$$\mathbf{Jhd}(3,4) = 36s234 * (\dot{\emptyset}_2+\dot{\emptyset}_3+\dot{\emptyset}_4)$$

The fourth row:

$$\mathbf{Jhd}(4,1) = 0$$

$$\mathbf{Jhd}(4,2) = 0$$

$$\mathbf{Jhd}(4,3) = 0$$

$$\mathbf{Jhd}(4,4) = 0$$

5.4. Inverse Kinematics

5.4.1. Joint Angles - A Geometric Approach. The inverse kinematic position analysis calculates the joint angles $\emptyset_1$, $\emptyset_2$, $\emptyset_3$, and $\emptyset_4$ given the desired position and orientation of the nozzle with respect to the global system.

An example of an inverse kinematic problem is to find the joint angles necessary to position and orientate the nozzle into the receiver. This requires that the position and

orientation of the receiver be known with respect to the global coordinate system. A coordinate system is attached to the receiver as shown in Figure 5-3 with the Z_r axis pointing out of the receiver. The rotation matrix ${}^0\mathbf{A}_r$ must be calculated from the receiver orientation.

With the assistance of Figure 5-3, the waist joint angle ($\emptyset_1$) of the base with respect to the vehicle can easily be determined from the global x and y position of the receiver.

Before the other joint angles can be determined, it is neccessary to determine the approach angle of the nozzle. The nozzle approach angle is the angle from the horizontal plane neccessary to put the nozzle directly into the receiver. As shown in Figure 5-3, the receiver centerline is orientated along the $-Z_r$ direction. The Z_r vector is projected onto the X_1, Y_1 plane since the nozzle orientation is constrained to move in this plane. The transformation matrix ${}^0\mathbf{A}_r$ relating the orientation of the receiver to the global coordinate system must by known and calculated. After the approach angle is known, calculate the position of coordinate system 3 with respect to coordinate system 0 and also calculate the position of coordinate system 3 with respect to coordinate system 1.

The shoulder joint angle ($\emptyset_2$) can be calculated as shown in Figure 5-3 using trigometric functions and relationships.

The elbow joint angle ($\emptyset_3$) can be calculated as shown in Figure 5-3 using trigometric functions and relationships.

The wrist joint angle ($\emptyset_4$) can easily be determined as shown in Figure 5-3.

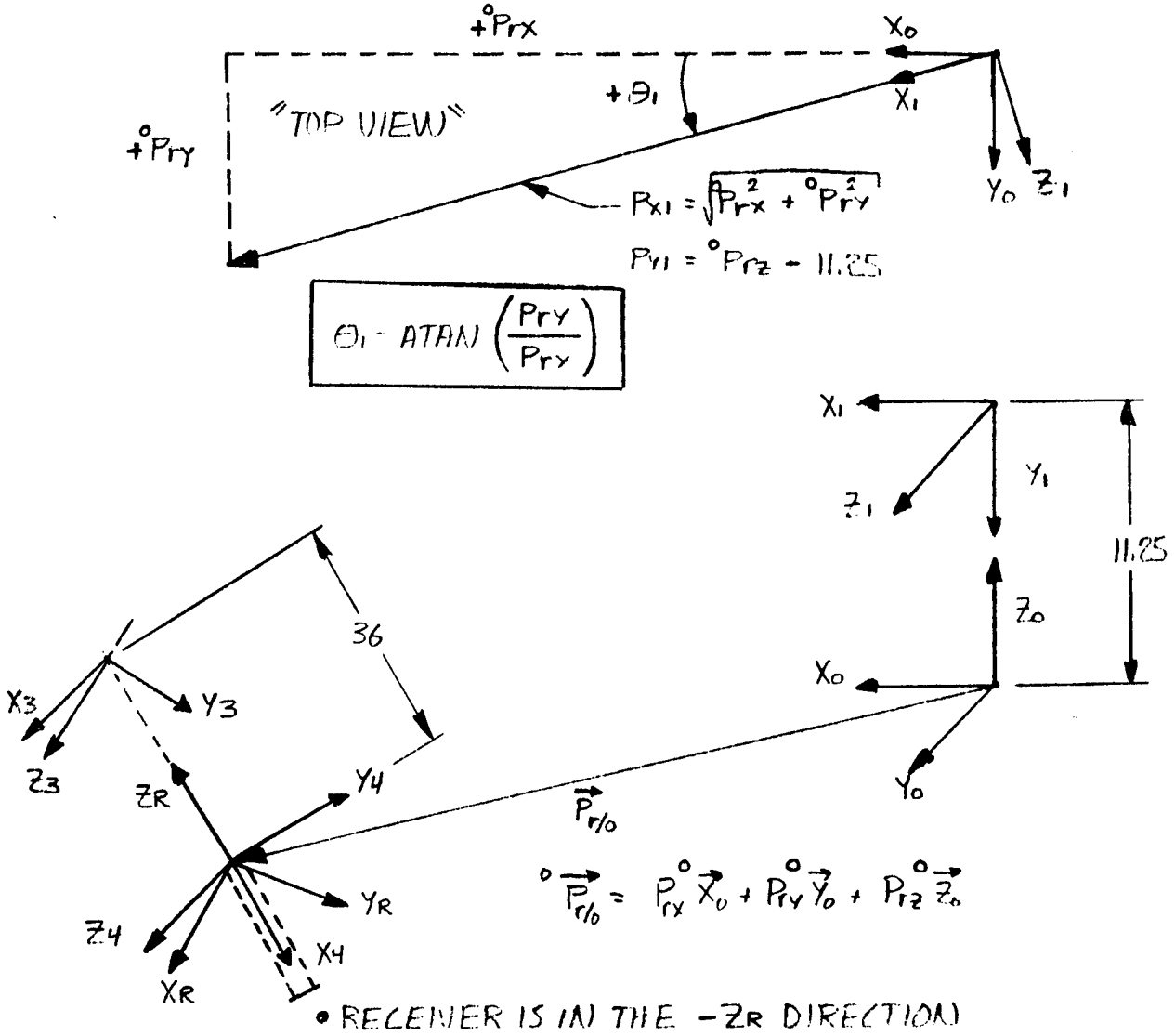
5.4.2. Joint Velocities. The inverse kinematic velocity analysis calculates the joint angle velocities given the nozzle velocity in Cartesian space. From the kinematic velocity analysis the following equation relating the joint velocities to cartesian velocities was developed:

$$\dot{\mathbf{y}} = \mathbf{Jh}\ \dot{\mathbf{q}}$$

The joint angle velocities are easily found by:

$$\dot{\mathbf{q}} = \mathbf{Jh}^{-1}\ \dot{\mathbf{y}}$$

5.4.3. Joint Accelerations. The inverse kinematic acceleration analysis calculates the joint angle accelerations given the nozzle acceleration in Cartesian space. From the kinematic acceleration analysis the following equation relating the joint accelerations to cartesian accelerations was developed:



• RECEIVER IS IN THE $-Z_R$ DIRECTION

• PROJECT Z_R INTO THE X_1, Y_1 PLANE AND CALCULATE THE APPROACH ANGLE. THE POSITION AND ORIENTATION OF THE RECEIVER MUST BE KNOWN.

PERFORM: $${}^1R_0\,{}^0R_R \begin{pmatrix} 0 \\ 0 \\ -Z_R \end{pmatrix} = ({}^0R_1)^T ({}^0R_R) \begin{pmatrix} 0 \\ 0 \\ -1 \end{pmatrix} = {}^1R_R \begin{pmatrix} 0 \\ 0 \\ -1 \end{pmatrix} = \begin{bmatrix} 0 & 0 & -a_{13} \\ 0 & 0 & -a_{23} \\ 0 & 0 & X \end{bmatrix}$$

$${}^1\theta_A = \text{ATAN}\left(\frac{-a_{23}}{-a_{13}}\right)$$

• DEFINE ${}^1A_4$ AS A ROTATION ${}^1\theta_A$ ABOUT THE Z_1 AXIS; $\text{ROT}(Z_1, {}^1\theta_A)$

$${}^1R_4 = \begin{bmatrix} \cos({}^1\theta_A) & -\sin({}^1\theta_A) & 0 \\ \sin({}^1\theta_A) & \cos({}^1\theta_A) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

• CALCULATE THE POSITION OF COORDINATE SYSTEM 3

$${}^0P_{3/0} = {}^0P_{R/0} - {}^0R_1\,{}^1R_4 \begin{pmatrix} -36 \\ 0 \\ 0 \end{pmatrix} = \begin{bmatrix} P_{Rx} \\ P_{RY} \\ P_{RZ} \end{bmatrix} - \begin{bmatrix} C\theta_1 C\theta_A & X & X \\ S\theta_1 C\theta_A & X & X \\ -S\theta_A & X & X \end{bmatrix} \begin{bmatrix} 36 \\ 0 \\ 0 \end{bmatrix}$$

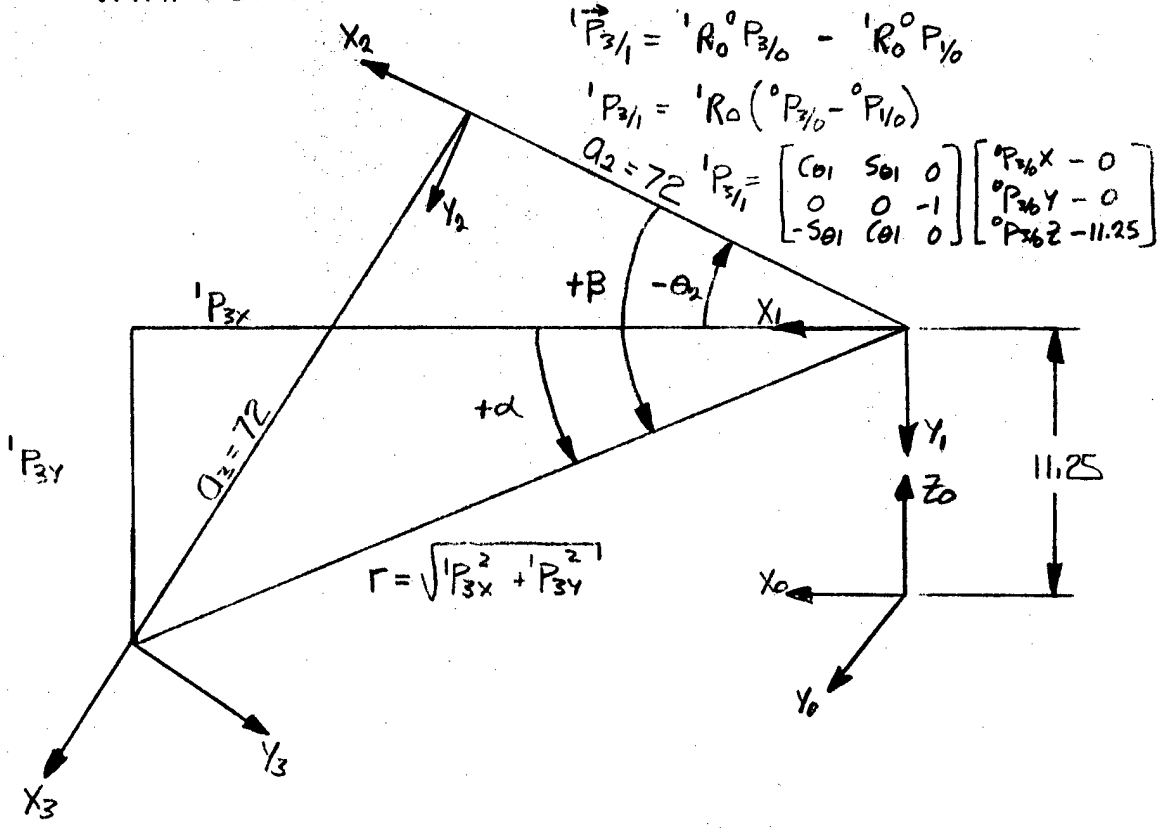
Figure 5-3. Inverse Kinematics - A Geometric Approach

- CALCULATE THE POSITION OF COORDINATE SYSTEM 3 WITH RESPECT TO COORDINATE SYSTEM 1

$${}^1\vec{P}_{3/1} = {}^1R_0\,{}^0P_{3/0} - {}^1R_0\,{}^0P_{1/0}$$

$${}^1P_{3/1} = {}^1R_0\left({}^0P_{3/0} - {}^0P_{1/0}\right)$$

$$a_2 = 72 \qquad {}^1P_{3/1} = \begin{bmatrix} C\theta_1 & S\theta_1 & 0 \\ 0 & 0 & -1 \\ -S\theta_1 & C\theta_1 & 0 \end{bmatrix} \begin{bmatrix} {}^0P_{3/0}X - 0 \\ {}^0P_{3/0}Y - 0 \\ {}^0P_{3/0}Z - 11.25 \end{bmatrix}$$



- $\text{SIN}\,\alpha = \dfrac{{}^1P_{3Y}}{r} \qquad \text{COS}\,\alpha = \dfrac{{}^1P_{3X}}{r}$

- $\text{COS}\,\beta = \dfrac{-a_3^2 + r^2 + a_2^2}{2 \cdot r \cdot a_2}$ "LAW OF COSINES" $\qquad \text{SIN}\,\beta = \sqrt{1 - \text{COS}^2\alpha}$

- $-\theta_2 = \beta - \alpha$

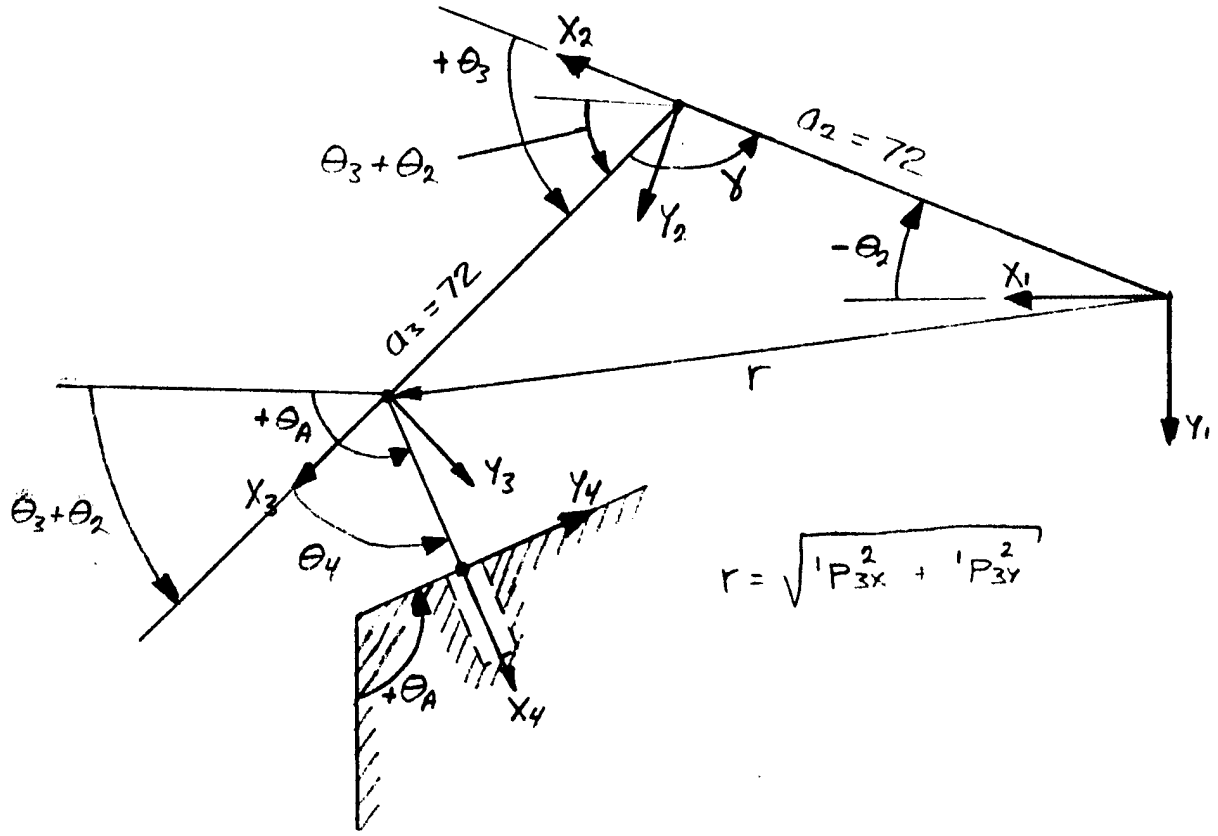
  $\theta_2 = \alpha - \beta$

- $\text{SIN}\,\theta_2 = \text{SIN}\,(\alpha - \beta) = \text{SIN}\,\alpha\,\text{COS}\,\beta - \text{COS}\,\alpha\,\text{SIN}\,\beta$

  $\text{COS}\,\theta_2 = \text{COS}\,(\alpha - \beta) = \text{COS}\,\alpha\,\text{COS}\,\beta + \text{SIN}\,\alpha\,\text{SIN}\,\beta$

$$\boxed{\theta_2 = \text{ATAN}\left(\frac{\text{SIN}\,\theta_2}{\text{COS}\,\theta_2}\right)}$$

Figure 5-3. Inverse Kinematics - A Geometric Approach (Continued)



- $\cos\gamma = \dfrac{-r^2 + a_2^2 + a_3^2}{2 \cdot a_2 \cdot a_3}$ $\qquad \sin\gamma = \sqrt{1 - \cos^2\gamma}$

- $\gamma = \text{ATAN}\left(\dfrac{\sin\gamma}{\cos\gamma}\right)$

$$\boxed{\theta_3 = \pi - \gamma}$$

- $\theta_2 + \theta_3 + \theta_4 = \theta_A$

$$\boxed{\theta_4 = \theta_A - \theta_3 - \theta_2}$$

Figure 5-3. Inverse Kinematics - A Geometric Approach (Continued)

$$\ddot{y} = \dot{Jh}\,\dot{q} + Jh\,\ddot{q}$$

The joint angle acclerations are easily found by:

$$\ddot{q} = Jh^{-1}\,(\,\ddot{y} - \dot{Jh}\,\dot{q}\,)$$

5.5. Dynamics

5.5.1. Introduction and Review. The following introduction and review of robot arm dynamics was taken from the book titled "Robotics: Control, Sensing, Vision, and Intelligence" by Fu, Gonzalez, and Lee.

"Robot arm dynamics deals with the mathematical formulation of the equations of robot arm motion. The dynamic equations of motion of a manipulator are a set of mathematical equations describing the dynamic behavior of the manipulator. Such equations are useful for computer simulation of the robot arm motion, the design of suitable control equations for a robot arm, and the evaluation of the kinematic design and structure of a robot arm. In this section we shall concentrate on the formulation, characteristics, and properties of the dynamic equations of motion suitable for control purposes. The purpose of manipulator control is to maintain the dynamic response of a computer based manipulator in accordance with some prespecified system performance and desired goals. In general, the dynamic responce of a manipulator directly depends on the efficiency of the control algorithms and the the dynamic model of the manipulator. The control problem consists of obtaining dynamic models of the physical robot arm system and then specifing corresponding control laws or strategies to achieve the desired system response and performance. This section deals mainly with the former part of the manipulator control problem; that is modeling and evaluating the dynamical properties and behavior of computer controlled robots.

The actual dynamic model of a robot arm can be obtained from known physical laws such as the laws of newtonian mechanics and lagrangian mechanics. This leads to the development of the dynamic equations of motion for the various articulated joints of the manipulator in terms of specified geometric and inertial parameters of the links. Conventional approaches like the Lagrange-Euler (L-E) and Newton-Euler (N-E) formulations could then be applied systematically to develop the actual robot arm motion equations. Various forms of robot arm motion equations describing the rigid body robot arm dynamics are obtained from these two formulations, such as Uicker's Lagrange-Euler equations

(Uicker [1965], Bejczy [1974]), Hollerbach's Recursive-Lagrange (R-L) equations (Hollerbach [1980]), Luh's Newton-Euler equations (Luh et al. [1980a]), and Lee's generalized d'Alembert (G-D) equations (Lee et al. [1983]). These motion equations are "equivalent" to each other in the sense that they describe the dynamic behavior of the same physical robot manipulator. However the structure of these equations may differ as they are obtained for various reasons and purposes. Some are obtained to achieve fast computation time in evaluating the nominal joint torques in servoing a manipulator, other are obtained to facilitate control analysis and synthesis, and still others are obtained to improve computer simulation of robot motion.

The derivation of the dynamic model of a manipulator based on the L-E formulation is simple and systematic. Assuming rigid body motion, the resulting equations of motion, excluding the dynamics of elctronic (or hydraulic) control devices, backlash, and gear friction, are a set of second-order coupled nonlinear differential equations. ... The L-E equations of motion provide explicit state equations for robot dynamics and can be utilized to analyze and design advanced joint-variable space control strategies. To a lesser extent, they are being used to solve for the forward dynamics problem, that is, given the desired torques/forces, the dynamic equations are used to solve for the joint accelerations which are then integrated to solve for the generalized coordinates and their velocities; or for the inverse dynamics problem, that is, given the desired generalized coordinates and their first two time derivatives, the generalized forces/torques are computed. ... Unfortunately, the computaton ... requires a fair amount of arithmetic operations. Thus, the L-E equations are very difficult to utilize for real-time control purposes unless they are simplified."[10]

The Lagrange Euler equation is:

$$\frac{d}{dt}\left[\frac{\partial L}{\partial \dot{q}_i}\right] - \frac{\partial L}{\partial q_i} = \tau_i \qquad i = 1, 2, \cdots, n$$

where L = Kinetic Energy - Potential Engergy (K - P)
K = total kinetic energy
P = total potential energy
q = generalized coordinates
$\dot{q}$ = first time derivative of the generalized coordinate
τ_i = generalized force or torque applied to the system

"From the Lagrange Euler equation, one is required to properly choose a set of generalized coordinates which completely describe the location (position and orienation) of a system with respect to a reference coordinate frame. For a simple manipulator with rotary-prismatic joints, various sets of generalized coordinates are available to describe the manipulator. However, since the angular positions of the joints are readily available because they can be measured by potentiometers or encoders or other sensing devices, they provide a natural correspondence with the generalized coordinates. ... Thus, in the case of a rotary joint, $q_i = \emptyset_i$, the joint angle span of the joint; whereas for a prismatic joint, $q_i = d_i$, the distance traveled by the joint."[11]

5.5.2. Potential Energy. The potential energy of link "i" is given by:

$$P_i = m_i * g * h_i$$

where: m_i = mass of link "i"
g = gravity
h_i = height of link "i"

Note that the potential energy is independent of joint velocity and is only dependent on joint position (q). The total potential energy is simply the sum of each link's potential energy.

$$P = \sum_{i=1}^{n} P_i$$

Since the potential energy is independent of joint velocity and is only dependent on joint position (q) the Lagrange equation can by rewritten as:

$$\frac{d}{dt}\left[\frac{\partial L}{\partial \dot{q}_i}\right] - \frac{\partial K}{\partial q_i} = \tau_i + \tau_{g_i} \qquad i = 1, 2, \cdots, n$$

where: $\tau_{g_i} = -\frac{\partial P}{\partial q}$

5.5.3. Mass Matrix. The mass matrix for link "i" is represented as:

$$\mathbf{M}_i = \begin{bmatrix} m_i & 0 & 0 \\ 0 & m_i & 0 \\ 0 & 0 & m_i \end{bmatrix}$$

5.5.4. Inertial Tensor. The inertial tensor, consisting of the moments of inertia and products of inertia for link "i", is a constant if expressed in a fixed body frame, but time varing if expressed in some other coordinate system. The body fixed inertial tensor for link "i" is represented as:

$$\Gamma_i = \begin{bmatrix} I_{xx} & -I_{xy} & -I_{xz} \\ -I_{yx} & I_{yy} & -I_{yz} \\ -I_{zx} & -I_{zy} & I_{zz} \end{bmatrix}$$

or using the radius of gyration:

$$k_{xy}^2 = I_{xy} / m_i$$

the inertia tensor can be rewritten as:

$$\Gamma_i = m_i * \begin{bmatrix} k_{xx}^2 & k_{xy}^2 & k_{xz}^2 \\ k_{yx}^2 & k_{yy}^2 & k_{yz}^2 \\ k_{zx}^2 & k_{zy}^2 & k_{zz}^2 \end{bmatrix}$$

5.5.5. Kinetic Energy. Chasle's Theorem states "The most general displacement of a rigid body is equivalent to a translation of some point in the body plus a rotation about an axis through that point".[12]

For convience the point of translation will be at the CG.

The kinetic energy of a link is composed of two parts; kinetic energy due to translational velocity at the cg, and kinetic energy due to a rotational velocity about an axis through the cg. Thus, the kinetic energy for link "i" is:

$$KE_{cg} = 1/2 * \mathbf{V}_{cg}^T \mathbf{M} \mathbf{V}_{cg} + 1/2 * \mathbf{W}^T \Gamma_{cg} \mathbf{W}$$

where: $\mathbf{V}_{cg}$ = velocity of link "i" at the cg
$\mathbf{W}$ = angular velocity of link "i"
Γ_{cg} = Moment and Product of Inertia for link "i" at the CG

Since the coordinate system attached to link "i" is usually not at the link's cg, it is necessary to rewrite the kinetic energy to express the kinetic energy using velocities about the link's coordinate system. First define a vector from the body fixed coordinate system to the cg:

$$\overline{O_iC} = \begin{bmatrix} C_x \\ C_y \\ C_z \end{bmatrix}$$

or in matrix form as:

$$\mathbf{C} = \begin{bmatrix} 0 & -C_z & C_y \\ C_z & 0 & -C_x \\ -C_y & C_x & 0 \end{bmatrix}$$

Using Figure 5-4, the kinetic energy is rewritten using the velocity at coordinate system "i", which is fixed to link "i". The angular velocity at coordinate system "i" is the same as the angular velocity at the CG since the angular velocity vector is a "free" vector. The resulting kinetic energy expression is given below:

$$KE_i = 1/2 * m_i \mathbf{V}_i^T \mathbf{V}_i + m_i \mathbf{V}_i^T \mathbf{C}_i^T \mathbf{W}_i + 1/2 * \mathbf{W}_i^T \Gamma_i \mathbf{W}_i$$

If we redefine $\mathbf{V}_i$ as:

$$\mathbf{V}_i = \begin{bmatrix} V_i \\ \\ W_i \end{bmatrix}$$

V_i velocity at coordinate system "i"

W_i angular velocity of link "i"

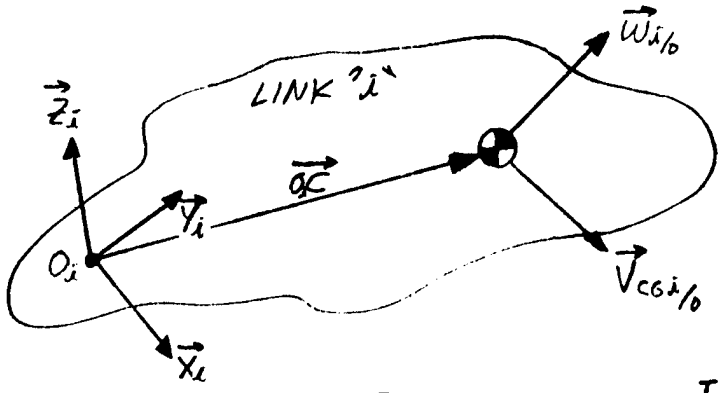
and define $\mathbf{U}_i$ as:

$$\mathbf{U}_i = \begin{bmatrix} m_i \mathbf{I}_3 & m_i \mathbf{C}_i^T \\ \\ m_i \mathbf{C}_i & \Gamma_i \end{bmatrix}$$

then it can be easily verified that the kinetic energy of link "i" is:

$$KE_i = 1/2 * \mathbf{V}_i^T \mathbf{U}_i \mathbf{V}_i$$

The total kinetic energy is found by summing the individual kinetic energy of each link. The total kinetic energy for a system with "n" links is given by:



KINETIC ENERGY AT C.G.: $KE = \frac{1}{2} V_{CG/0}^T m_i V_{CG/0} + \frac{1}{2} W_{i/0}^T \Gamma_{CG} W_{i/0}$

VELOCITY OF CG: $\vec{V}_{CG/0} = \vec{V}_{i/0} + \vec{\omega}_{i/0} \times \vec{c}_{CG/i}$

$$\vec{V}_{CG/0} = \vec{V}_{i/0} + \tilde{c}^T \vec{\omega}_{i/0}$$

SUBSITUTE: $KE = \frac{1}{2} [\vec{V}_{i/0} + \tilde{c}^T \vec{\omega}_{i/0}]^T m_i [\vec{V}_{i/0} + \tilde{c}^T \vec{\omega}_{i/0}] + \frac{1}{2} \vec{\omega}_{i/0}^T \Gamma_{CG} \vec{\omega}_{i/0}$

$= \frac{1}{2} [\vec{V}_{i/0}^T + \vec{\omega}_{i/0}^T \tilde{c}] m_i [\vec{V}_{i/0} + \tilde{c}^T \vec{\omega}_{i/0}] + \frac{1}{2} \vec{\omega}_{i/0}^T \Gamma_{CG} \vec{\omega}_{i/0}$

$= \frac{1}{2} m_i [\vec{V}_{i/0}^T \vec{V}_{i/0} + \vec{V}_{i/0}^T \tilde{c}^T \vec{\omega}_{i/0} + \vec{\omega}_{i/0}^T \tilde{c} \vec{V}_{i/0} + \vec{\omega}_{i/0}^T \tilde{c} \tilde{c}^T \vec{\omega}_{i/0}] + \frac{1}{2} \vec{\omega}_{i/0}^T \Gamma_{CG} \vec{\omega}_{i/0}$

$= \frac{1}{2} m_i [\vec{V}_{i/0}^T \vec{V}_{i/0} + \vec{V}_{i/0}^T \tilde{c}^T \vec{\omega}_{i/0} + (\tilde{c}^T \vec{\omega}_{i/0})^T \vec{V}_{i/0}] + \frac{1}{2} m_i \vec{\omega}_{i/0}^T \tilde{c} \tilde{c}^T \vec{\omega}_{i/0} + \frac{1}{2} \vec{\omega}_{i/0}^T \Gamma_{CG} \vec{\omega}_{i/0}$

$= \frac{1}{2} m_i [\vec{V}_{i/0}^T \vec{V}_{i/0} + \underbrace{\vec{V}_{i/0}^T \tilde{c}^T \vec{\omega}_{i/0}}_{\text{scaler "s"}} + \underbrace{(\vec{V}_{i/0}^T \tilde{c}^T \vec{\omega}_{i/0})^T}_{\text{scaler "s}^T\text{"}}] + \frac{1}{2} \vec{\omega}_{i/0}^T \underbrace{[\Gamma_{CG} + m_i \tilde{c} \tilde{c}^T]}_{\text{Parallel-Axis Theorem}} \vec{\omega}_{i/0}$

$s + s^T = 2s$

$$KE = \frac{1}{2} m_i \vec{V}_{i/0}^T \vec{V}_{i/0} + m_i \vec{V}_{i/0}^T \tilde{c}^T \vec{\omega}_{i/0} + \frac{1}{2} \vec{\omega}_{i/0}^T [\Gamma_i] \vec{\omega}_{i/0}$$

Inertia with respect to the "i" coordinate system

$$\tilde{c} = \begin{bmatrix} 0 & -c_z & c_y \\ c_z & 0 & -c_x \\ -c_y & c_x & 0 \end{bmatrix}$$

$$\vec{\omega} \times \vec{c} = \tilde{c}^T \omega = -\tilde{c} \omega = \begin{bmatrix} 0 & c_z & -c_y \\ -c_z & 0 & c_x \\ c_y & -c_x & 0 \end{bmatrix} \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} = \begin{bmatrix} \omega_y c_z - \omega_z c_y \\ -\omega_x c_z + \omega_z c_x \\ \omega_x c_y - \omega_y c_x \end{bmatrix}$$

Figure 5-4. Kinetic Energy at Coordinate System "i"

$$KE = 1/2 * \sum_{i=1}^{n} \mathbf{V}_i^T \mathbf{U}_i \mathbf{V}_i$$

Note that $\mathbf{V}_i$ is in cartesian coordinates, however, we want to use generalized coordinates. We can convert cartesian coordinates to generalized coordinate by using a subjacobian matrix $\mathbf{J}_i$ as follows:

$$\mathbf{V}_i = \begin{bmatrix} V_i \\ \\ W_i \end{bmatrix} = \mathbf{J}_i \dot{\mathbf{q}}$$

The derivation and a method to calculate the subjacobian matrix $\mathbf{J}_i$ is given in Figure 5-5.

Subsituting the above into the expression for the total kinetic energy and combining terms, the kinetic energy can easily be rewritten as:

$$KE = 1/2 * \dot{\mathbf{q}}^T \mathbf{W} \dot{\mathbf{q}}$$

where $\mathbf{W} = \sum_{i=1}^{n} \mathbf{J}_i^T \mathbf{U}_i \mathbf{J}_i$

The Inertial matrix $\mathbf{W}$ is symetric and positive definite.

5.5.6. Lagrange Equation. In a paper by Gu and Loh titled "Dynamic Model for Industrial Robots Based on a Compact Lagrangian Formulation" the following dynamic equation was formulated using the Lagrange Equation along with the Kinetic Energy and Potential Engergy equations previously derived:[13]

$$\mathbf{W} \ddot{\mathbf{q}} + (\bar{\mathbf{W}}^T - 1/2\bar{\mathbf{W}}) \dot{\mathbf{q}} = \mathcal{T} + \mathcal{T}_g$$

where:

$$\bar{\mathbf{W}} = \begin{bmatrix} \dot{\mathbf{q}}^T \partial \mathbf{W}/\partial q_1 \\ \vdots \\ \dot{\mathbf{q}}^T \partial \mathbf{W}/\partial q_n \end{bmatrix}$$

For link 1 of the robotic refueler arm, the mass matrix, the inertia tensor, the subjacobian matrix, and the $\mathbf{W}_1$

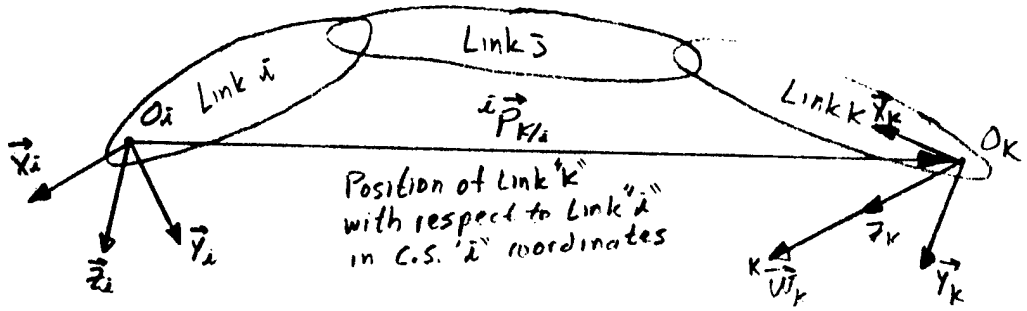
DERIVATION AND METHOD FOR CALCULATING THE SUBJACOBIAN MATRIX

GOAL: $\begin{pmatrix} {}^i\vec{V} \\ {}^i\vec{\omega} \end{pmatrix} = J_i \dot{q}$

${}^i\vec{V}$ = Velocity of CS "i"

${}^i\vec{\omega}$ = Angular Velocity of CS "i"

All Velocities are projected unto the i^{th} Coordinate System



$$ {}^K A_i = \begin{bmatrix} {}^K R_i & {}^K\vec{P}_{i/k} \\ 0 & 1 \end{bmatrix} . $$

Angular Velocity of Link "k" ${}^K\vec{W}_K$ is along the $\vec{Z}_K$ axis.

$$ {}^i A_K \begin{bmatrix} {}^i R_K & {}^i\vec{P}_{K/i} \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} {}^i\vec{X}_K & {}^i\vec{Y}_K & {}^i\vec{Z}_K & {}^i\vec{P}_{K/i} \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

Angular Velocity: ${}^K\vec{\omega}_K = \dot{\theta}_K {}^K\vec{Z}_K$

$\dot{\theta}_K \rightarrow$ Angular Velocity of Joint "K"

$$ {}^i\vec{\omega}_K = {}^i R_K {}^K\vec{\omega}_K = \dot{\theta}_K {}^i R_K {}^K\vec{Z}_K = \dot{\theta}_K {}^i\vec{Z}_K $$

$\rightarrow 3^{rd}$ Column of Homogeneous Transformation Matrix

Velocity: ${}^i V_i = {}^i\omega_K \times (-{}^i P_{K/i})$

$$ {}^i V_i = {}^i P_{K/i} \times {}^i\omega_K = {}^i P_{K/i} \times {}^i Z_K \; [\dot{\theta}_K] $$

$\rightarrow 3^{rd}$ Column Vector

$\rightarrow 4^{th}$ Column Vector

For each Column of J_i

$$ Y_i = \begin{cases} \begin{bmatrix} {}^i P_{K/i} \times {}^i Z_K \\ {}^i Z_K \end{bmatrix} & \text{for a Revolute} \\ \begin{bmatrix} {}^i Z_K \\ 0 \end{bmatrix} & \text{for a Prismatic} \end{cases} $$

Figure 5-5. Derivation and Method for Calculating the Subjacobian Matrix

Method)

1. Calculate: ${}^{0}A_0, {}^{0}A_1, \cdots {}^{0}A_{i-1}$

2. Extract: ${}^{0}\vec{z}_k \rightarrow 3^{rd}$ Column Vector
   ${}^{i}P_{k/i} \rightarrow 4^{th}$ Column Vector

3. Perform: ${}^{i}P_{k/i} \times {}^{0}z_k$

4. Construct:

$$J = \begin{bmatrix} {}^{0}P_{0/i} \times {}^{0}z_0 & {}^{0}P_{1/i} \times {}^{0}z_1 & \cdots & {}^{0}P_{i-1/i} \times {}^{0}z_{j-1} \\ {}^{0}z_0 & {}^{0}z_1 & \cdots & {}^{0}z_{j-1} \end{bmatrix}$$

If joint "k" is prismatic then replace column "k" with $\begin{bmatrix} {}^{0}z_k \\ 0 \end{bmatrix}$

Figure 5-5. Derivation and Method for Calculating the Subjacobian Matrix (Continued)

matrix are shown in Figure 5-6. For link 2 the mass matrix, the inertia tensor, the subjacobian matrix, and the W_2 matrix are shown in Figure 5-7. For link 3 the mass matrix, the inertia tensor, the subjacobian matrix, and the W_3 matrix are shown in Figure 5-8. For link 4 the mass matrix, the inertia tensor, the subjacobian matrix, and the W_4 matrix are shown in Figure 5-9. The total **W** matrix for the robotic arm is:

$$\mathbf{W} = \mathbf{W}_1 + \mathbf{W}_2 + \mathbf{W}_3 + \mathbf{W}_4$$

The calculations for computing the subjacobian matrix J_i and the W_i matrix of each link is given in Appendix A.

The compact Lagrangian formulation stated above provides a method for developing a control strategy for the robotic arm.

5.6. State Space Control Model

5.6.1. Nonlinear State Space Representation in Joint Coordinates. The nonlinear state space representation is given by:

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}) + \mathbf{B}(\mathbf{x})*\mathbf{u}$$
$$\mathbf{y} = \mathbf{h}(\mathbf{x}) = \mathbf{h}(\mathbf{q})$$

where, for the robotic arm, small "**x**" contains the state variables of joint positions and joint velocities as shown below:

$$\mathbf{x} = \begin{bmatrix} \mathbf{q} \\ \dot{\mathbf{q}} \end{bmatrix}$$

Joint Position

Joint Velocity

The time derivative of "**x**" is:

$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{\mathbf{q}} \\ \ddot{\mathbf{q}} \end{bmatrix}$$

Joint Velocity

Joint Acceleration

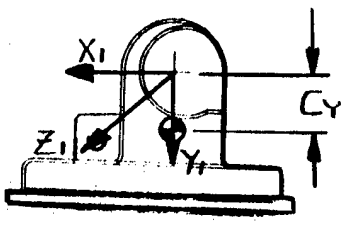
The output "**y**" is position and orientation of the nozzle:

$$\mathbf{y} = \begin{bmatrix} \mathbf{P} \\ \mathbf{O} \end{bmatrix}$$

Position

Orienatation

LINK 1
ROTATING BASE



$$C_1 = \begin{bmatrix} 0 & 0 & C_y \\ 0 & 0 & 0 \\ -C_y & 0 & 0 \end{bmatrix} \qquad M_1 = \begin{bmatrix} m_1 & 0 & 0 \\ 0 & m_1 & 0 \\ 0 & 0 & m_1 \end{bmatrix} \qquad \Gamma_1 = \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix}$$

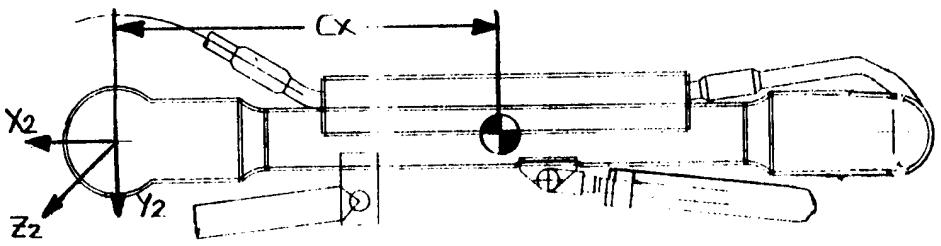
$$U_1 = \begin{bmatrix} m_1 & 0 & 0 & 0 & 0 & -C_y*m_1 \\ 0 & m_1 & 0 & 0 & 0 & 0 \\ 0 & 0 & m_1 & C_y*m_1 & 0 & 0 \\ 0 & 0 & C_y*m_1 & I_{xx} & 0 & 0 \\ 0 & 0 & 0 & 0 & I_{yy} & 0 \\ -C_y*m_1 & 0 & 0 & 0 & 0 & I_{zz} \end{bmatrix}$$

$$J_1 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$W_1 = \begin{bmatrix} I_{yy} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Figure 5-6. Rotating Base (Link 1) Subjacobian J_1 and W_1 Matrices

LINK 2
AFT ARM



$$C_2 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -C_x \\ 0 & C_x & 0 \end{bmatrix} \qquad M_2 = \begin{bmatrix} m_2 & 0 & 0 \\ 0 & m_2 & 0 \\ 0 & 0 & m_2 \end{bmatrix} \qquad \Gamma_2 = \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix}$$

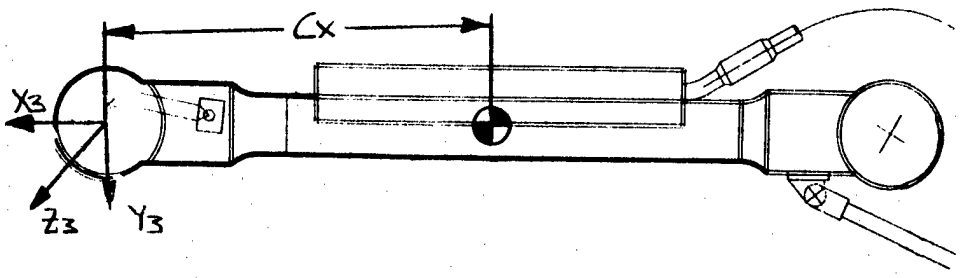
$$U_2 = \begin{bmatrix} m_2 & 0 & 0 & 0 & 0 & 0 \\ 0 & m_2 & 0 & 0 & 0 & C_x*m_2 \\ 0 & 0 & m_2 & 0 & -C_x*m_2 & 0 \\ 0 & 0 & 0 & I_{xx} & 0 & 0 \\ 0 & 0 & -C_x*m_2 & 0 & I_{yy} & 0 \\ 0 & C_x*m_2 & 0 & 0 & 0 & I_{zz} \end{bmatrix}$$

$$J_2 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 72 & 0 & 0 \\ 72*c2 & 0 & 0 & 0 \\ -s2 & 0 & 0 & 0 \\ -c2 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

$$W_2 = \begin{bmatrix} 72^2*c2^2m_2+2*72*c2^2C_xm_2+I_{xx}s2^2+I_{yy}c2^2 & & 0 & 0 \\ 0 & 72^2*m_2+2*72*C_xm_2+I_{zz} & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Figure 5-7. Aft Arm (Link 2) Subjacobian J_2 and W_2 Matrices

LINK 3
FORE ARM



$$C_3 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -C_x \\ 0 & C_x & 0 \end{bmatrix} \qquad M_3 = \begin{bmatrix} m_3 & 0 & 0 \\ 0 & m_3 & 0 \\ 0 & 0 & m_3 \end{bmatrix} \qquad \Gamma_3 = \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix}$$

$$U_3 = \begin{bmatrix} m_3 & 0 & 0 & 0 & 0 & 0 \\ 0 & m_3 & 0 & 0 & 0 & C_x*m_3 \\ 0 & 0 & m_3 & 0 & -C_x*m_3 & 0 \\ 0 & 0 & 0 & I_{xx} & 0 & 0 \\ 0 & 0 & -C_x*m_3 & 0 & I_{yy} & 0 \\ 0 & C_x*m_3 & 0 & 0 & 0 & I_{zz} \end{bmatrix}$$

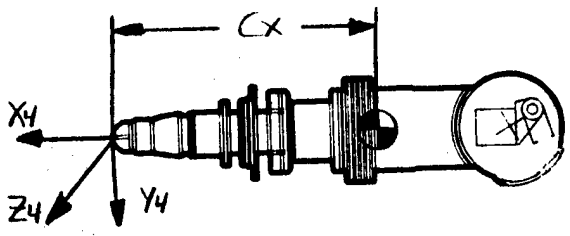
$$J_3 = \begin{bmatrix} 0 & 72*s3 & 0 & 0 \\ 0 & 72+72*c3 & 72 & 0 \\ 72*c23+72*c2 & 0 & 0 & 0 \\ -s23 & 0 & 0 & 0 \\ -c23 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \end{bmatrix}$$

Figure 5-8. Fore Arm (Link 3) Subjacobian J_3 and W_3 Matrices

$$W_3 = \begin{bmatrix} \begin{pmatrix} r_2^2 C_{23}^2 m_3 + (2)(r_2)^2 C_{23} C_2 m_3 \\ + r_2^2 C_2^2 m_3 + (2)(r_2) C_{23}^2 C_x m_3 \\ + (2)(r_2) C_{23} C_2 C_x m_3 \\ + I_{xx} S_{23}^2 + I_{yy} C_{23}^2 \end{pmatrix} & 0 & 0 & 0 \\ 0 & \begin{pmatrix} (2)(r_2)^2 m_3 + (2)(r_2)^2 C_3 m_3 \\ + (2)(r_2) C_x m_3 + I_{zz} \\ + (2)(r_2) C_3 C_x m_3 \end{pmatrix} & \begin{pmatrix} r_2^2 m_3 + r_2^2 C_3 m_3 \\ + (2)(r_2) C_x m_3 + r_2 C_3 C_x m_3 \\ + I_{zz} \end{pmatrix} & 0 \\ 0 & \begin{pmatrix} r_2^2 m_3 + r_2^2 C_3 m_3 \\ + (2)(r_2) C_x m_3 + r_2 C_3 C_x m_3 \\ + I_{zz} \end{pmatrix} & \begin{pmatrix} r_2^2 m_3 + (2)(r_2) C_x m_3 + I_{zz} \end{pmatrix} & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Figure 5-8. Fore Arm (Link 3) Subjacobian J_3 and W_3 Matrices (Continued)

LINK 4
NOZZLE



$$C_4 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -C_x \\ 0 & C_x & 0 \end{bmatrix} \qquad M_4 = \begin{bmatrix} m_4 & 0 & 0 \\ 0 & m_4 & 0 \\ 0 & 0 & m_4 \end{bmatrix} \qquad \Gamma_4 = \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix}$$

$$U_4 = \begin{bmatrix} m_4 & 0 & 0 & 0 & 0 & 0 \\ 0 & m_4 & 0 & 0 & 0 & C_x*m_4 \\ 0 & 0 & m_4 & 0 & -C_x*m_4 & 0 \\ 0 & 0 & 0 & I_{xx} & 0 & 0 \\ 0 & 0 & -C_x*m_4 & 0 & I_{yy} & 0 \\ 0 & C_x*m_4 & 0 & 0 & 0 & I_{zz} \end{bmatrix}$$

$$J_4 = \begin{bmatrix} 0 & 72*s4+72*s34 & 72*s4 & 0 \\ 0 & 36+72*c4+72*c34 & 36+72*c4 & 36 \\ 36*c234+72*c23+72*c2 & 0 & 0 & 0 \\ -s234 & 0 & 0 & 0 \\ -c234 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \end{bmatrix}$$

Figure 5-9. Nozzle (Link 4) Subjacobian J_4 and W_4 Matrices

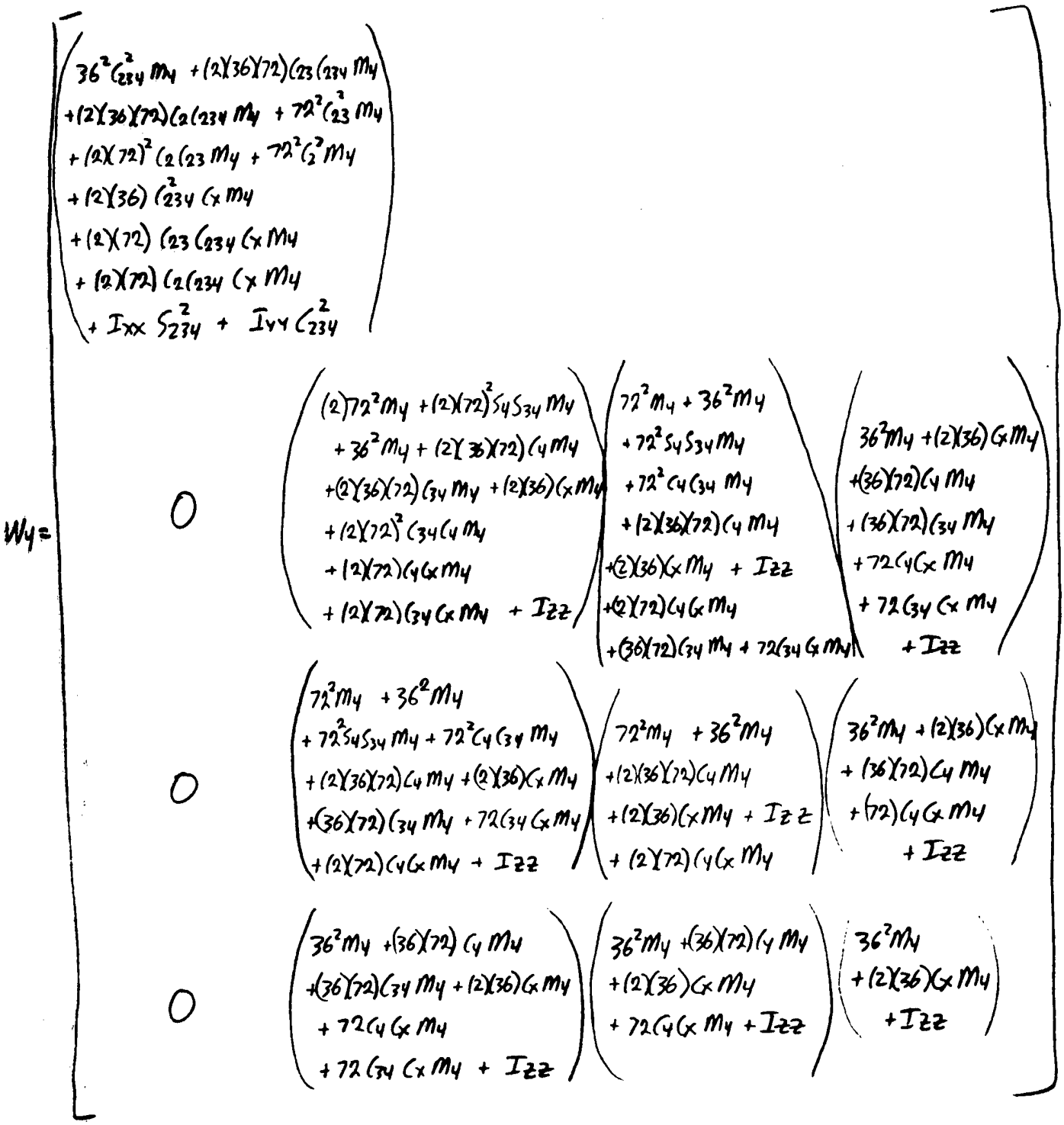


Figure 5-9. Nozzle (Link 4) Subjacobian J_4 and W_4 Matrices (Continued)

The input torque "**u**" includes both the dynamic torque and the gravitation torque $_g$:

$$\mathbf{u} = \begin{bmatrix} \mathcal{T}_1 + \mathcal{T}_{g1} \\ \vdots \\ \mathcal{T}_4 + \mathcal{T}_{g4} \end{bmatrix} \quad \begin{matrix} \text{Torque on joint 1} \\ \vdots \\ \text{Torque on joint 4} \end{matrix}$$

From the Lagrange equation the nonlinear state space model is:

$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{\mathbf{q}} \\ \ddot{\mathbf{q}} \end{bmatrix} = \begin{bmatrix} \dot{\mathbf{q}} \\ \mathbf{W}^{-1}[\,\mathbf{u} - (\,\bar{\mathbf{W}}^T - 1/2\dot{\bar{\mathbf{W}}}\,)\dot{\mathbf{q}}\,] \end{bmatrix}$$

$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{\mathbf{q}} \\ \mathbf{W}^{-1}(\,1/2\dot{\bar{\mathbf{W}}} - \bar{\mathbf{W}}^T\,)\dot{\mathbf{q}} \end{bmatrix} + \begin{bmatrix} 0 \\ \mathbf{W}^{-1} \end{bmatrix} \mathbf{u}$$

5.6.2. Linear State Space Representation in Global Coordinates. To control the global position and orientation of the nozzle (Coordinate System 4) the nonlinear state space representation given in joint coordinates must be transformed into a new set of state variable in global or base coordinates. The new state variables in global coordinates, designated as capital "**X**", is defined as:

$$\mathbf{X} = \begin{bmatrix} \mathbf{y} \\ \dot{\mathbf{y}} \end{bmatrix} \quad \begin{matrix} \text{Global Position of Nozzle} \\ \text{Global Velocity of Nozzle} \end{matrix}$$

The time derivative of capital "**X**" is:

$$\dot{\mathbf{X}} = \begin{bmatrix} \dot{\mathbf{y}} \\ \ddot{\mathbf{y}} \end{bmatrix} \quad \begin{matrix} \text{Global Velocity of Nozzle} \\ \text{Global Acceleration of Nozzle} \end{matrix}$$

The new output vector, designated as capital **Y**, contains the global position and orientation of the nozzle:

$$Y = \begin{bmatrix} P \\ O \end{bmatrix}$$

Position

Orienatation

The linear equations in state space are:

$$\dot{X} = AX + BV$$

$$Y = CX$$

where the new input $V = \ddot{y}$. The linear state space equations are equivalent to:

$$\dot{X} = \begin{bmatrix} y \\ \ddot{y} \end{bmatrix} = \begin{bmatrix} y \\ \dot{y} \end{bmatrix} = \begin{bmatrix} 0 & I \\ 0 & 0 \end{bmatrix} \begin{bmatrix} y \\ \dot{y} \end{bmatrix} + \begin{bmatrix} 0 \\ I \end{bmatrix} V$$

$$Y = \begin{bmatrix} I & 0 \end{bmatrix} \begin{bmatrix} y \\ \dot{y} \end{bmatrix}$$

The nonlinear transformation from joint coordinates to global or base coordinates is given by the robot arm kinematic equations. The transformation equations are:

$$y = h(q)$$

$$\dot{y} = Jh\,\dot{q}$$

$$\ddot{y} = \dot{Jh}\,\dot{q} + Jh\,\ddot{q}$$

The linear input vector V can be converted into joint torques by subsituting:

$$\ddot{q} = Jh^{-1}(\ddot{y} - \dot{Jh}\,\dot{q}) = Jh^{-1}(V - \dot{Jh}\,\dot{q})$$

into the Lagrange equation. The torque u can be written as:

$$B(X)\,V + \mathcal{C}(x) = u$$

where:

$$\alpha(x) = (\bar{W}^T - 1/2\bar{W})\dot{q} - W\, Jh^{-1}\dot{Jh}\,\dot{q}$$

$$B(x) = W\, Jh^{-1}$$

5.6.3. Global Proportional Derivative (PD) Control System. A global proportional derivative control block diagram for the robot is shown in Figure 5-10. The desired global position y_d is compared to the actual position y to determine the position error. The desired global velocity $\dot{y}_d$ is compared to the actual velocity $\dot{y}$ to determine the velocity error. The input vector V is the sum of the position error multiplied by a gain k1, the velocity error multiplied by a gain k2, and the desired global acceleration $\ddot{y}_d$. The equation, $B(X)\ V + \alpha(x)$, can then be used to calculate the required joint torques for controling the robot system.

5.6.4. Performance. Performance and stability of the control system depends upon the values selected for the position error gain k1 and the velocity error gain k2. Performance and stability is determined by the corresponding characteristic equation. The characteristic equation for the simple linear state space model is equivalent to:

$$\det [\ \lambda I - (A - B K)\ ] = \lambda^2 + k2*\lambda + k1$$

$$= \lambda^2 + 2*\xi*w_n*\lambda + w_n^2$$

The gain k2 is equivalent to $2*\xi*w_n$ and the gain k1 is equivalent w_n^2 where ξ is the damping ratio and w_n is the natural frequency. For critical damping, no overshoot, $\xi = 1$. The settling time is dependent upon both the damping ratio and the natural frequency.

5.7. Simulation of Global Position Controller

5.7.1. Computer Program. A computer program was written using the "C" language to simulate the robot refueler arm's kinematic, dynamic, and control equations. Appendix C contains all the source code for the simulation.

The source code was compiled using the "Optimizing C86 Compiler", version 2.3, by Computer Innovations, Inc.

Execution of the program begins with the routine called "main". This routine calls the routine "control". By studing the "control" routine, the reader can easily follow program exection.

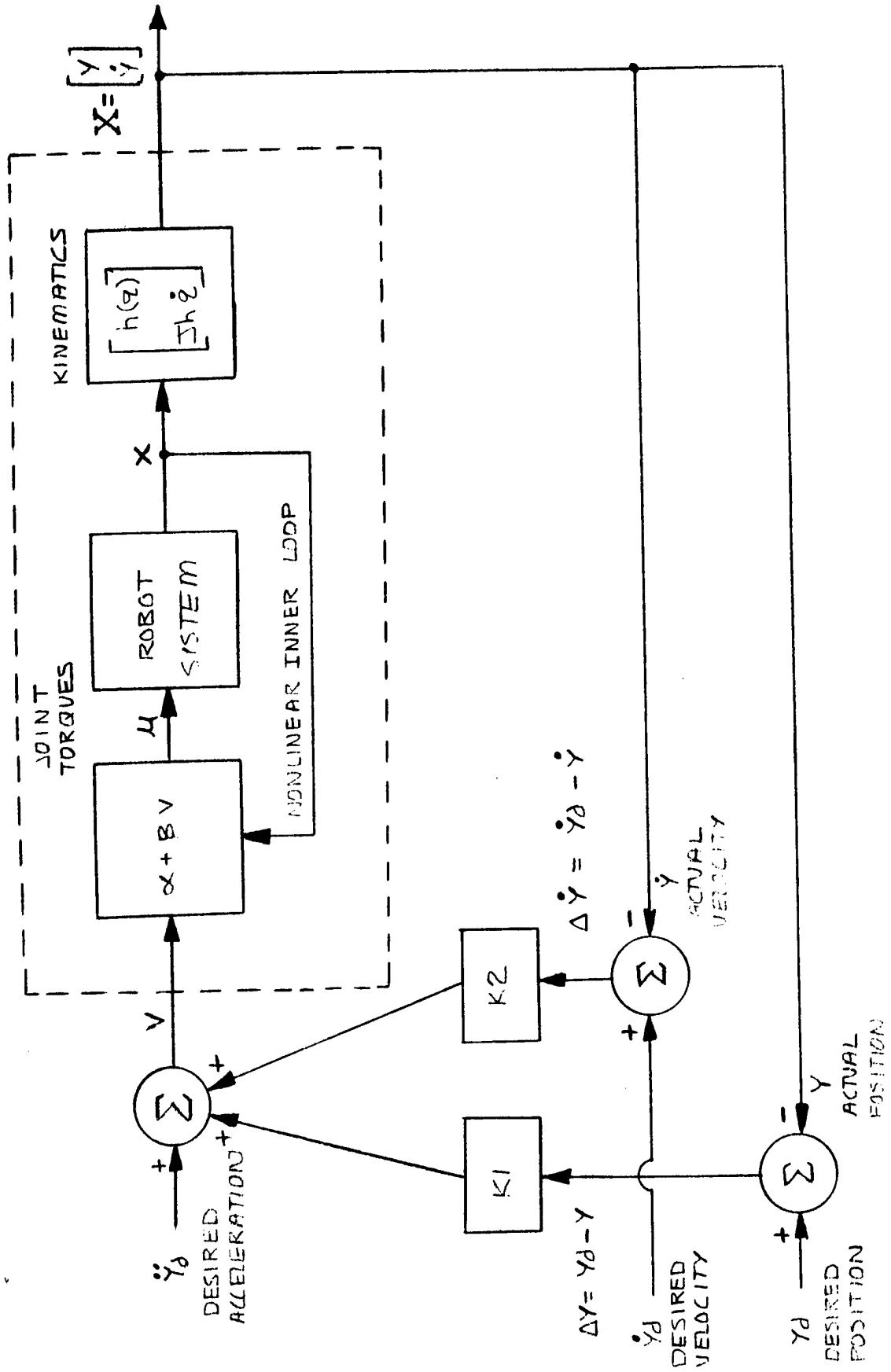


Figure 5-10. Global Proportional Derivative (PD) Control Block Diagram

5.7.2. Performance. The desired robot performance was chosen as follows:

- 0% Overshoot - Critical Damping - $\zeta = 1.0$
- Natural Frequency = 8.0 Hz.

To acheive the desired performance, gain values of K1 = 64.0 and K2 = 16.0 were chosen.

5.7.3. Simulation Results. The desired trajectory, which the contolled robot is suppose to follow, was chosen to be 72-inch radius circular path in the horizontal plane with a simultaneous 48-inch sinusoidal rise in elevation while maintaining a constant approach angle of 90 degrees. The equations which describe the desired trajectory path are as follows:

$$\mathbf{Y\_d} = \begin{bmatrix} 72.0 * \cos (pi*t/10.0) \\ 72.0 * \sin (pi*t/10.0) \\ -24.75 + 48.0 * \sin (pi*t/10.0) \\ pi/2.0 \end{bmatrix}$$

The initial conditions in the global X, Y, and Z direction were purposely offset from the desired trajectory by 12.0 inches and the nozzle orientation was offset by 12 degrees to evaluate the performance of the control system against disturbances.

The results of the simulation are given in Figure 5-11 through Figure 5-16. Figure 5-11 shows the nozzle global X position as a function of time while figure 5-12 shows the nozzle global Y position and Figure 5-13 shows the global Z position as a function of time. Figure 5-14 shows the nozzle approach angle as a function of time. In the four figures the desired trajectory is illustrated by the solid curve and the actual position of the nozzle is illustrated by the dotted curve.

Figure 5-15 shows the nozzle position error in each direction and Figure 5-16 shows the nozzle orientation error. Initially the position error is 12.0 inches and the orientation error is 12.0 degrees as a result of the initial conditions. After one second the position error is nearly zero and the nozzle follows the desired tractory as expected.

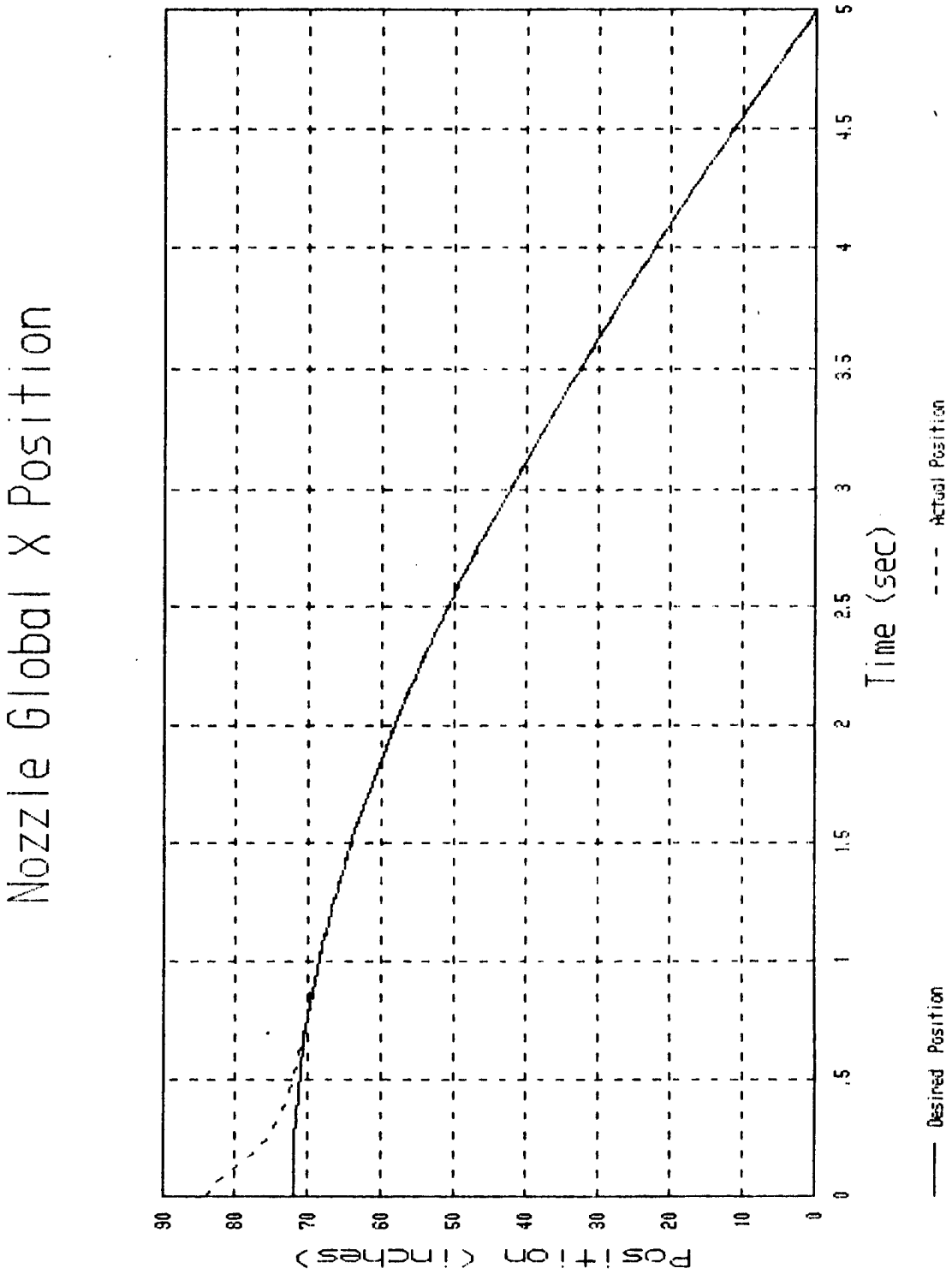


Figure 5-11. Nozzle Desired Trajectory and Actual Global X Position

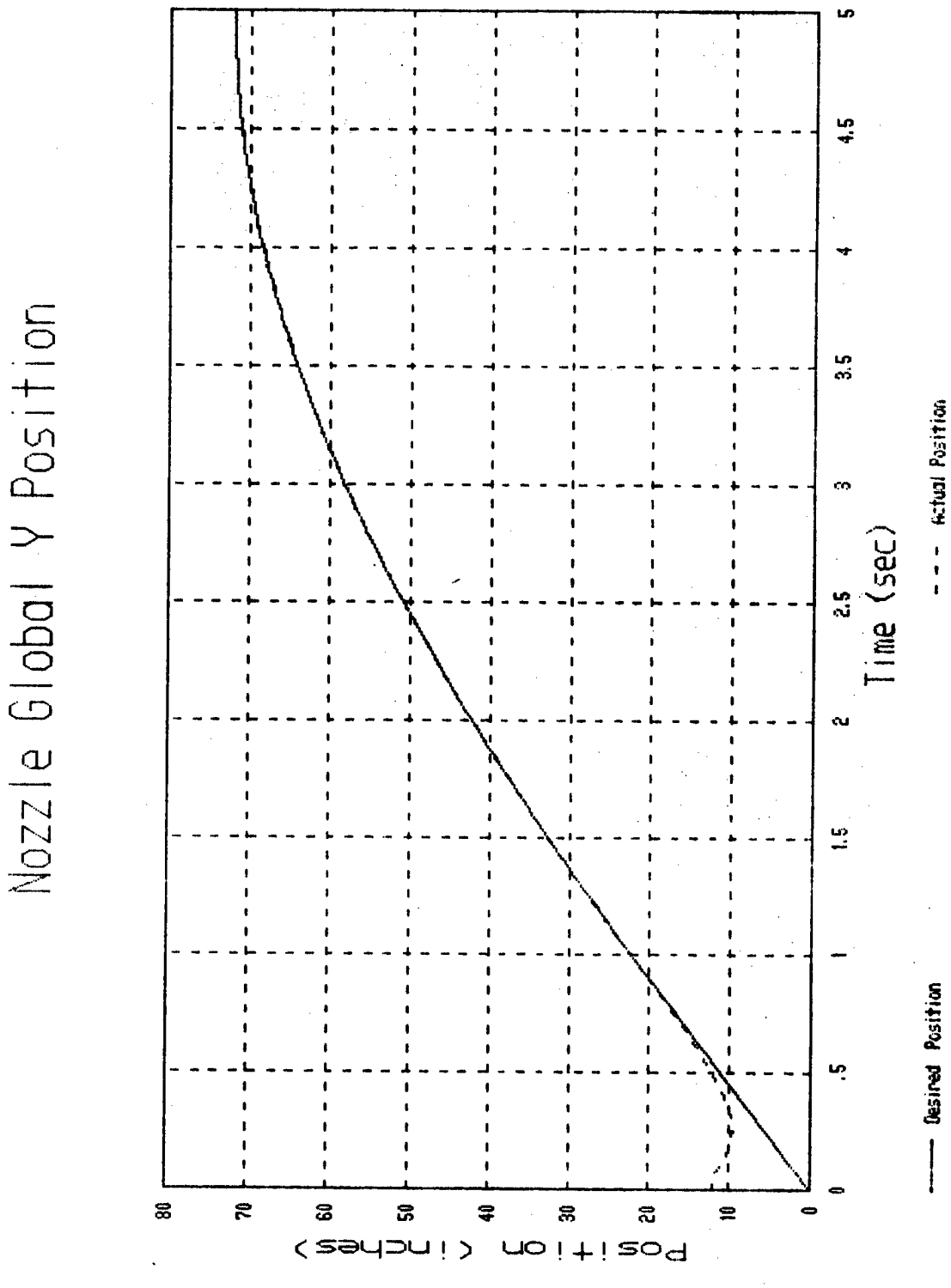


Figure 5-12. Nozzle Desired Trajectory and Actual Global Y Position

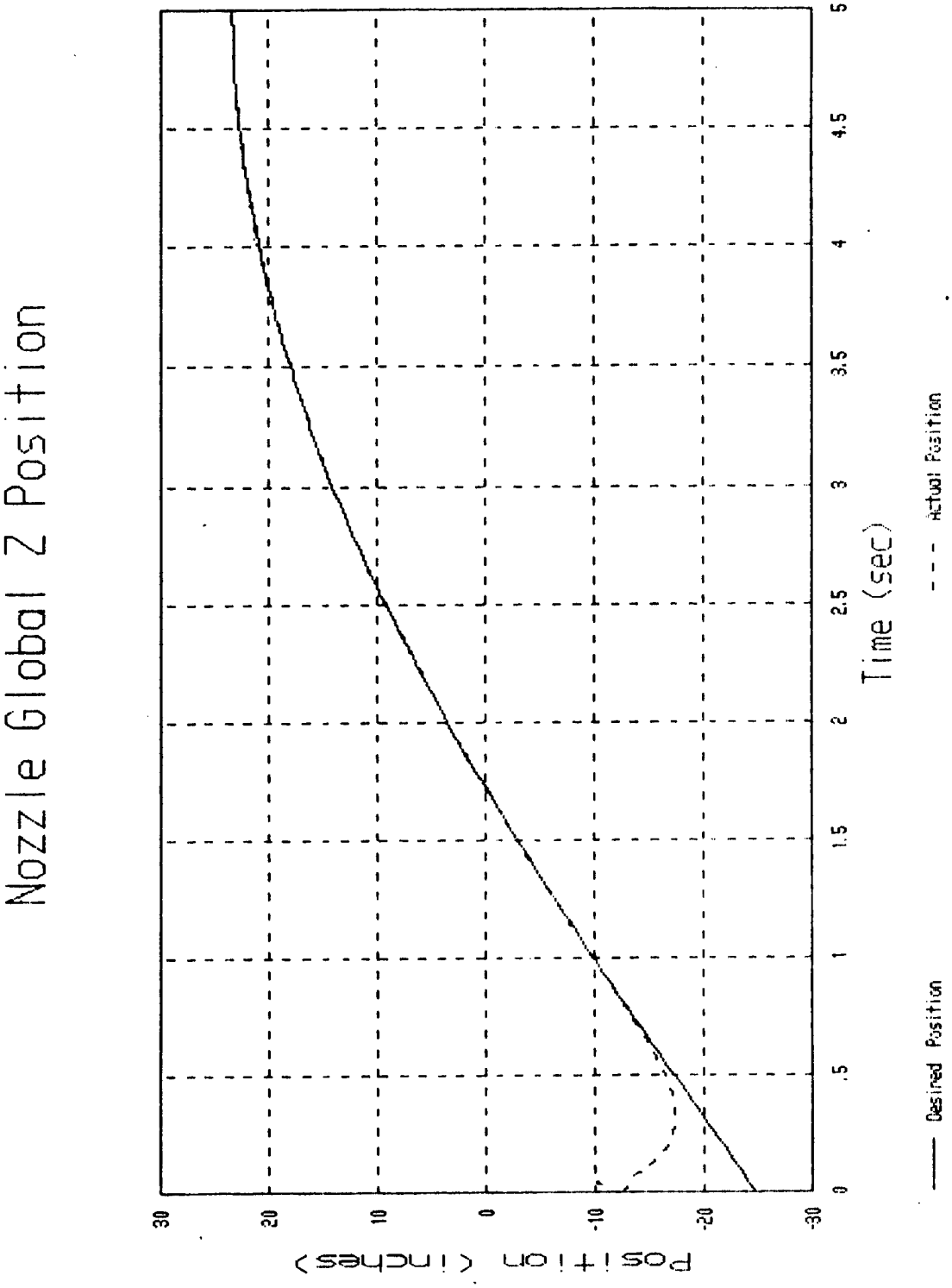


Figure 5-13. Nozzle Desired Trajectory and Actual Global Z Position

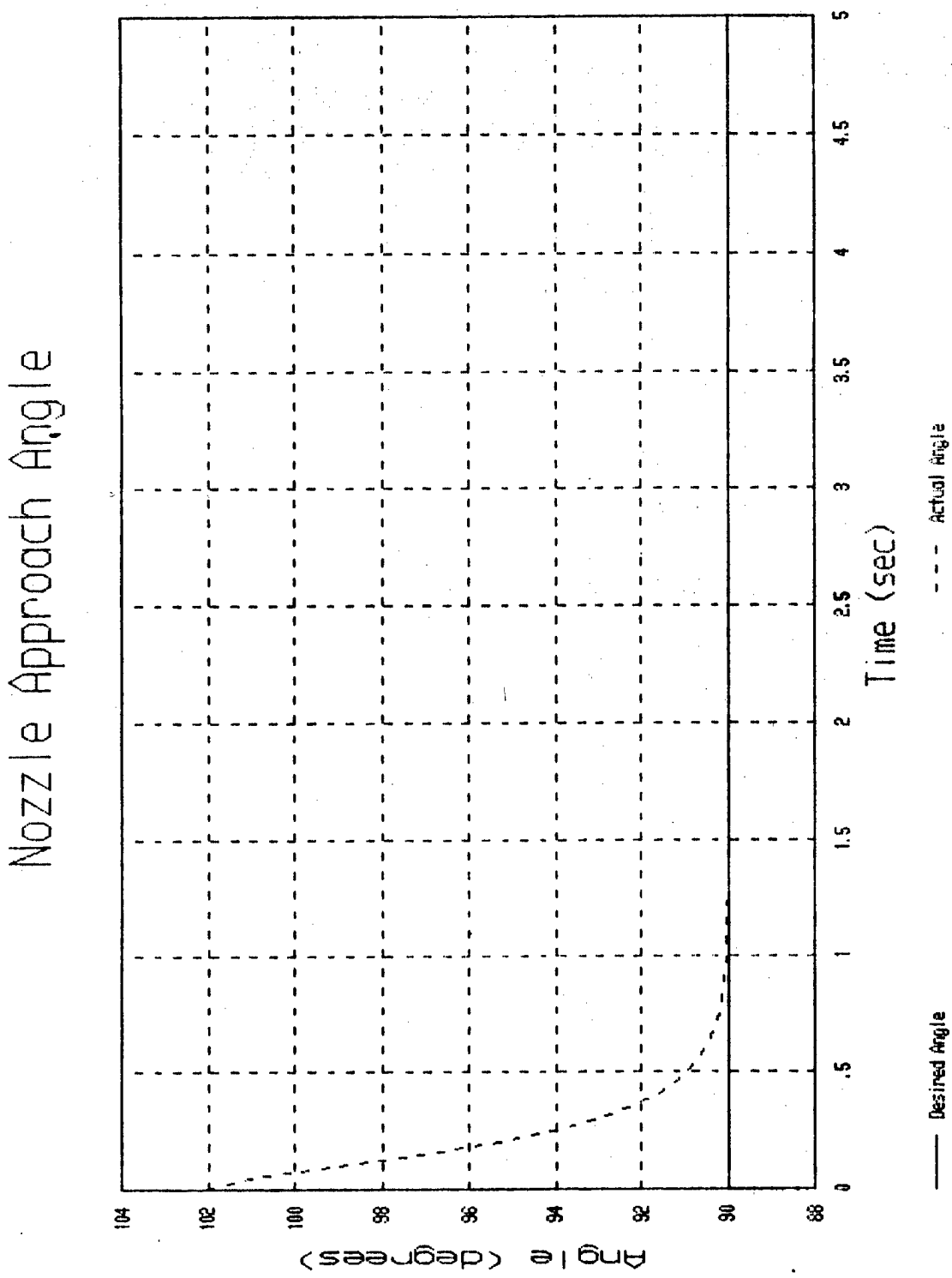


Figure 5-14. Nozzle Desired Trajectory and Actual Approach Angle

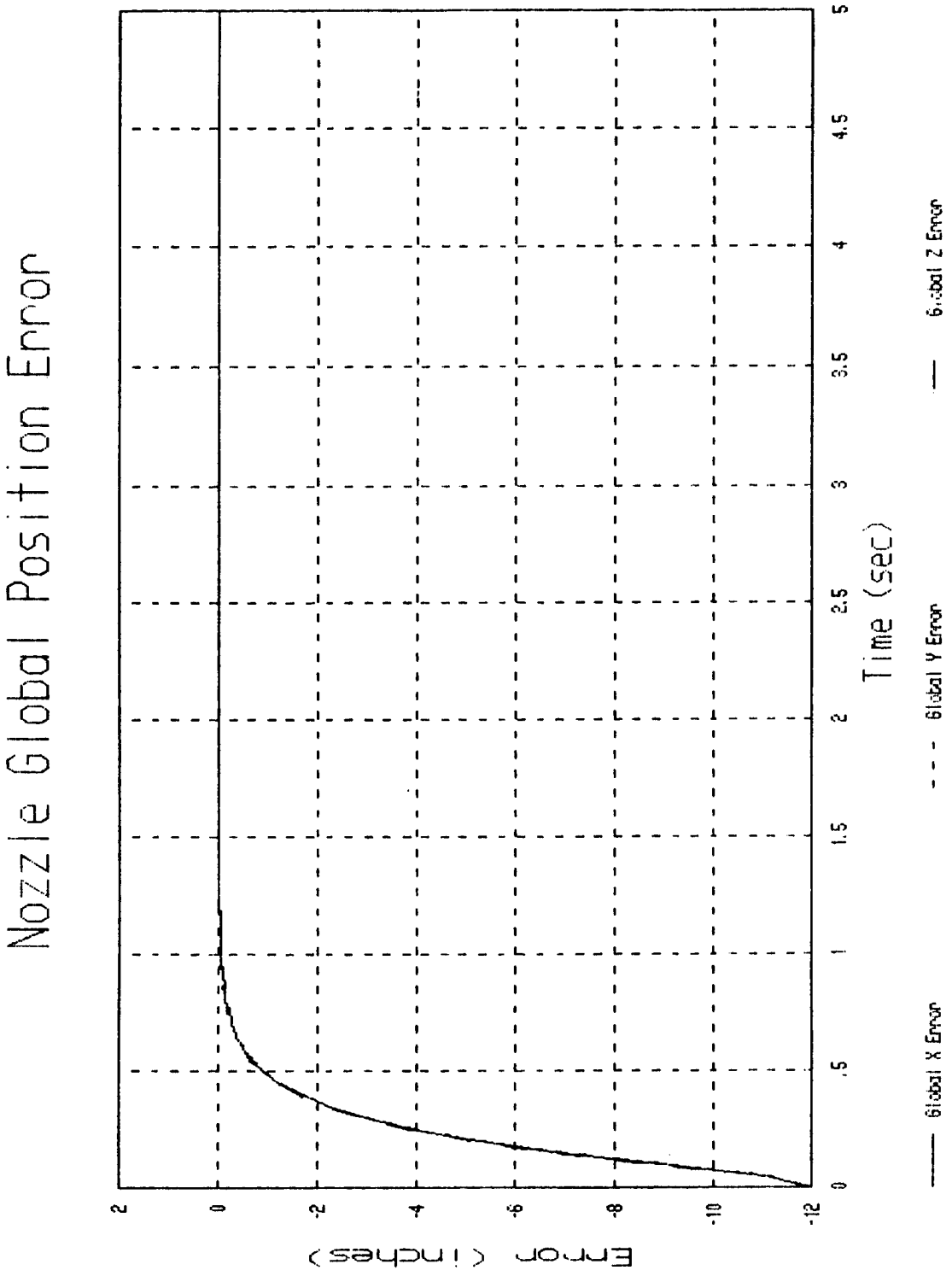


Figure 5-15. Nozzle Global Position Error

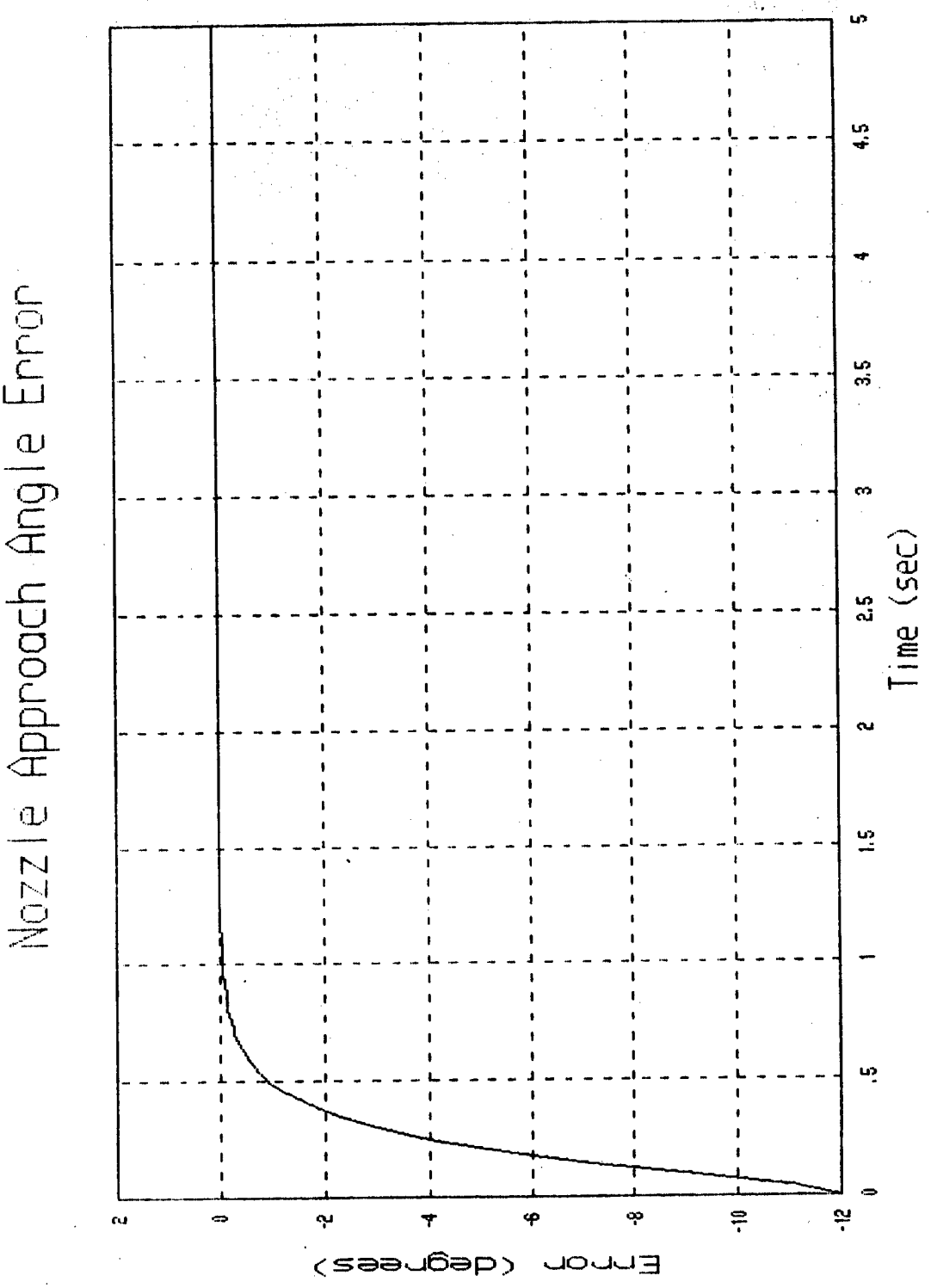


Figure 5-16. Nozzle Approach Angle Error

LIST OF REFERENCES

1 Gound, Dennis and Bearden, James, "Customer Test of Refueler Demonstrator", TRADOC TRMS Number 87-0000781, US Army Armor Center and Fort Knox, ATTN: ATZK-DS, Fort Knox, KY, p. 1-1 (1987)

2 Gound, Dennis and Bearden, James, "Customer Test of Refueler Demonstrator", TRADOC TRMS Number 87-0000781, US Army Armor Center and Fort Knox, ATTN: ATZK-DS, Fort Knox, KY, p. 1-1 (1987)

3 Gound, Dennis and Bearden, James, "Customer Test of Refueler Demonstrator", TRADOC TRMS Number 87-0000781, US Army Armor Center and Fort Knox, ATTN: ATZK-DS, Fort Knox, KY, p. 1-2 (1987)

4 Gound, Dennis and Bearden, James, "Customer Test of Refueler Demonstrator", TRADOC TRMS Number 87-0000781, US Army Armor Center and Fort Knox, ATTN: ATZK-DS, Fort Knox, KY, p. 1-2 (1987)

5 Gound, Dennis and Bearden, James, "Customer Test of Refueler Demonstrator", TRADOC TRMS Number 87-0000781, US Army Armor Center and Fort Knox, ATTN: ATZK-DS, Fort Knox, KY, p. 1-3 (1987)

6 Gound, Dennis and Bearden, James, "Customer Test of Refueler Demonstrator", TRADOC TRMS Number 87-0000781, US Army Armor Center and Fort Knox, ATTN: ATZK-DS, Fort Knox, KY, p. 1-3 (1987)

7 Fu, K. S., Gonzalez, R. C., Lee, C. S. G., "Robotics: Control, Sensing, Vision, and Intelligence", McGraw-Hill, Inc., p. 36 (1987)

8 Fu, K. S., Gonzalez, R. C., Lee, C. S. G., "Robotics: Control, Sensing, Vision, and Intelligence", McGraw-Hill, Inc., p. 27-28 (1987)

9 Fu, K. S., Gonzalez, R. C., Lee, C. S. G., "Robotics: Control, Sensing, Vision, and Intelligence", McGraw-Hill, Inc., p. 30-31 (1987)

10 Fu, K. S., Gonzalez, R. C., Lee, C. S. G., "Robotics: Control, Sensing, Vision, and Intelligence", McGraw-Hill, Inc., p. 82-83 (1987)

11 Fu, K. S., Gonzalez, R. C., Lee, C. S. G., "Robotics: Control, Sensing, Vision, and Intelligence", McGraw-Hill, Inc., p. 85 (1987)

12 Greenwood, Donald T., "Principles of Dynamics", Prentice-Hall, Inc., p. 319 (1965)

13 Gu, You-Liang, Loh, Nan K., "Dynamic Model for Industrial Robots Based on a Compact Lagrangian Formulation", Proceedings on the 24th IEEE Conference on Decision and Control, December, (1985)

SELECTED BIBLIOGRAPHY

Brady, Michael, Hollerbach, John M., Johnson, Timothy L., Lozano-Perez, Tomas, Mason, Matthew T., "Robot Motion: Planning and Control," Second Printing, Massachusetts Institute of Technology (MIT) Press, Cambridge, MA (1983)

Fu, K. S., Gonzalez, R. C., Lee, C.S.G., "Robotics: Control, Sensing, Vision, and Intelligence," McGraw-Hill, Inc., New York, NY (1987)

Greenwood, Donald T. "Principles of Dynamics," Prentice-Hall, Inc., Englewood Cliffs, NJ (1965)

Gu, You-Liang, Loh, Nan K., "Dynamic Model for Industrial Robots Based on a Compact Lagrangian Formulation," Proceedings on the 24th Conference on Decision and Control, December (1985)

"Optimizing C86 Compiler," Version 2.3, Computer Innovations, Inc., Trinton Falls, NJ (1985)

APPENDIX A

LINK SUBJACOBIAN, $\mathbf{W}$, AND $\overline{\mathbf{W}}$ MATRIX CALCULATIONS

Calculate J_1:

$$^0A_1 = \begin{bmatrix} C_1 & 0 & -S_1 & 0 \\ S_1 & 0 & C_1 & 0 \\ 0 & -1 & 0 & 11.25 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$^1A_0 = \begin{bmatrix} C_1 & S_1 & 0 & 0 \\ 0 & 0 & -1 & 11.25 \\ -S_1 & C_1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$^1P_{0/1} \times {}^1Z_0 = \begin{bmatrix} 0 \\ 11.25 \\ 0 \end{bmatrix} \times \begin{bmatrix} 0 \\ -1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

$$J_1 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Calculate W_1 $\qquad W_1 = J_1^T M_1 J_1$

$$W_1 = J_1^T \begin{bmatrix} m_1 & 0 & 0 & 0 & 0 & -C_Y m_1 \\ 0 & m_1 & 0 & 0 & 0 & 0 \\ 0 & 0 & m_1 & C_Y m_1 & 0 & 0 \\ 0 & 0 & C_Y m_1 & I_{xx} & 0 & 0 \\ 0 & 0 & 0 & 0 & I_{yy} & 0 \\ -C_Y m_1 & 0 & 0 & 0 & 0 & I_{zz} \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$W_1 = \begin{bmatrix} 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ -I_{yy} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$W_1 = \begin{bmatrix} I_{yy} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Calculate $\frac{\partial W_1}{\partial q_1}, \frac{\partial W_2}{\partial q_2}, \frac{\partial W_3}{\partial q_3}, \frac{\partial W_4}{\partial q_4},$

$$\frac{\partial W_1}{\partial q_1} = \frac{\partial W_2}{\partial q_2} = \frac{\partial W_3}{\partial q_3} = \frac{\partial W_4}{\partial q_4} = [0]$$

Calculate I_2:

$$ {}^1A_2 = \begin{bmatrix} C_2 & -S_2 & 0 & 72C_2 \\ S_2 & C_2 & 0 & 72S_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^2A_1 = \begin{bmatrix} C_2 & S_2 & 0 & -72C_2^2 - 72S_2^2 \\ -S_2 & C_2 & 0 & +72C_2S_2 - 72C_2S_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} C_2 & S_2 & 0 & -72 \\ -S_2 & C_2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^0A_2 = {}^0A_1\,{}^1A_2 = \begin{bmatrix} C_1 & 0 & -S_1 & 0 \\ S_1 & 0 & C_1 & 0 \\ 0 & -1 & 0 & 11.25 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} C_2 & -S_2 & 0 & 72C_2 \\ S_2 & C_2 & 0 & 72S_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^0A_2 = \begin{bmatrix} C_1C_2 & -C_1S_2 & -S_1 & 72C_1C_2 \\ S_1C_2 & -S_1S_2 & C_1 & 72S_1C_2 \\ -S_2 & -C_2 & 0 & -72S_2 + 11.25 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$72S_2C_2 - 72S_2C_2$

$-72C_2^2(C_1^2 + S_1^2)$

$$ {}^2A_0 = \begin{bmatrix} C_1C_2 & S_1C_2 & -S_2 & -72C_1^2C_2^2 - 72S_1^2C_2^2 - 72S_2^2 + 11.25\,S_2 \\ -C_1S_2 & -S_1S_2 & -C_2 & 72C_1^2S_2C_2 + 72S_1^2S_2C_2 - 72S_2C_2 + 11.25C_2 \\ -S_1 & C_1 & 0 & 72S_1C_1C_2 - 72S_1C_1C_2 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^2A_0 = \begin{bmatrix} C_1C_2 & S_1C_2 & -S_2 & -72 + 11.25\,S_2 \\ -C_1S_2 & -S_1S_2 & -C_2 & 11.25C_2 \\ -S_1 & C_1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^{2}P_{0/2} \times {}^{2}\hat{z}_0 = \begin{bmatrix} -72 + 11.25 S_2 \\ 11.25 C_2 \\ 0 \end{bmatrix} \times \begin{bmatrix} -S_2 \\ -C_2 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ +72C_2 - 11.25 S_2 C_2 + 11.25 C_2 S_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 72C_2 \end{bmatrix} $$

$$ {}^{2}P_{1/2} \times {}^{2}\hat{z}_1 = \begin{bmatrix} -72 \\ 0 \\ 0 \end{bmatrix} \times \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ +72 \\ 0 \end{bmatrix} $$

$$ J_2 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 72 & 0 & 0 \\ 72C_2 & 0 & 0 & 0 \\ -S_2 & 0 & 0 & 0 \\ -C_2 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} $$

Calculate W_2 $\qquad W_2 = J_2^T U_2 J_2$

$$W_2 = J_2^T \begin{bmatrix} M_2 & 0 & 0 & 0 & 0 & 0 \\ 0 & M_2 & 0 & 0 & 0 & C_x M_2 \\ 0 & 0 & M_2 & 0 & -C_x M_2 & 0 \\ 0 & 0 & 0 & I_{xx} & 0 & 0 \\ 0 & 0 & -C_x M_2 & 0 & I_{yy} & 0 \\ 0 & C_x M_2 & 0 & 0 & 0 & I_{zz} \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 72 & 0 & 0 \\ 72 C_2 & 0 & 0 & 0 \\ -S_2 & 0 & 0 & 0 \\ -C_2 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

$$W_2 = \begin{bmatrix} 0 & 0 & 72C_2 & -S_2 & -C_2 & 0 \\ 0 & 72 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 72M_2 + C_x M_2 & 0 & 0 \\ 72C_2 M_2 + C_2 C_x M_2 & 0 & 0 & 0 \\ -I_{xx} S_2 & 0 & 0 & 0 \\ -72 C_2 C_x M_2 - I_{yy} C_2 & 0 & 0 & 0 \\ 0 & 72 C_x M_2 + I_{zz} & 0 & 0 \end{bmatrix}$$

$$W_2 = \begin{bmatrix} 72^2 C_2^2 M_2 + 72 C_2^2 C_x M_2 + I_{xx} S_2^2 + 72 C_2^2 C_x M_2 + I_{yy} C_2^2 & 0 & 0 & 0 \\ 0 & 72^2 M_2 + 72 C_x M_2 + 72 C_x M_2 + I_{zz} & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$W_2 = \begin{bmatrix} 72^2 C_2^2 M_2 + (2)(72) C_2^2 C_x M_2 + I_{xx} S_2^2 + I_{yy} C_2^2 & 0 & 0 & 0 \\ 0 & 72^2 M_2 + (2)(72) C_x M_2 + I_{zz} & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Calculate $\frac{\partial W_2}{\partial q_1}$, $\frac{\partial W_2}{\partial q_2}$, $\frac{\partial W_2}{\partial q_3}$, $\frac{\partial W_2}{\partial q_4}$

$$\frac{\partial W_2}{\partial q_1} = \frac{\partial W_2}{\partial q_3} = \frac{\partial W_2}{\partial q_4} = [0]$$

$$\frac{\partial W_2}{\partial q_2} = \begin{bmatrix} -(2)(72)^2 C_2 S_2 M_2 - (4)(72) C_2 S_2 C_x M_2 + (2) I_{xx} S_2 C_2 - (2) I_{yy} C_2 S_2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\frac{\partial W_2}{\partial q_2} = \begin{bmatrix} \frac{\partial W_2}{\partial q_2}(1,1) & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Calculate J_3:

$$^2A_3 = \begin{bmatrix} C_3 & -S_3 & 0 & r_2C_3 \\ S_3 & C_3 & 0 & r_2S_3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$^3A_2 = \begin{bmatrix} C_3 & S_3 & 0 & -r_2C_3^2 - r_2S_3^2 \\ -S_3 & C_3 & 0 & r_2S_3C_3 - r_2S_3C_3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} C_3 & S_3 & 0 & -r_2 \\ -S_3 & C_3 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$^1A_3 = {}^1A_2\,{}^2A_3 = \begin{bmatrix} C_2 & -S_2 & 0 & r_2C_2 \\ S_2 & C_2 & 0 & r_2S_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} C_3 & -S_3 & 0 & r_2C_3 \\ S_3 & C_3 & 0 & r_2S_3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$^1A_3 = \begin{bmatrix} C_2C_3 - S_2S_3 & -C_2S_3 - S_2C_3 & 0 & r_2C_2C_3 - r_2S_2S_3 + r_2C_2 \\ S_2C_3 + C_2S_3 & -S_2S_3 + C_2C_3 & 0 & r_2S_2C_3 + r_2C_2S_3 + r_2S_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$^1A_3 = \begin{bmatrix} C_{23} & -S_{23} & 0 & r_2C_{23} + r_2C_2 \\ S_{23} & C_{23} & 0 & r_2S_{23} + r_2S_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$^3A_1 = \begin{bmatrix} C_{23} & S_{23} & 0 & -r_2C_{23}^2 - r_2C_2C_{23} - r_2S_{23}^2 - r_2S_2S_{23} \\ -S_{23} & C_{23} & 0 & +r_2C_{23}S_{23} + r_2C_2S_{23} - r_2S_{23}C_{23} - r_2S_2C_{23} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$ {}^{3}A_1 = \begin{bmatrix} C_{23} & S_{23} & 0 & -72 - 72C_3 \\ -S_{23} & C_{23} & 0 & 72S_3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^{0}A_3 = {}^{0}A_1\,{}^{1}A_3 = \begin{bmatrix} C_1 & 0 & -S_1 & 0 \\ S_1 & 0 & C_1 & 0 \\ 0 & -1 & 0 & 11.25 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} C_{23} & -S_{23} & 0 & 72C_{23} + 72C_2 \\ S_{23} & C_{23} & 0 & 72S_{23} + 72S_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^{0}A_3 = \begin{bmatrix} C_1C_{23} & -C_1S_{23} & -S_1 & 72C_1C_{23} + 72C_1C_2 \\ S_1C_{23} & -S_1S_{23} & C_1 & 72S_1C_{23} + 72S_1C_2 \\ -S_{23} & -C_{23} & 0 & -72S_{23} - 72S_2 + 1125 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^{3}A_0 = \begin{bmatrix} C_1C_{23} & S_1C_{23} & -S_{23} & -72C_1^2C_{23}^2 - 72C_1^2C_2C_{23} - 72S_1^2C_{23}^2 - 72S_1^2C_2C_{23} - 72S_{23}^2 - 72S_2S_{23} + 11.25S_{23} \\ -C_1S_{23} & -S_1S_{23} & -C_{23} & 72C_1^2C_{23}S_{23} + 72C_1^2C_2S_{23} + 72S_1^2C_{23}S_{23} + 72S_1^2C_2S_{23} - 72S_{23}C_{23} - 72S_2C_{23} + 11.25C_{23} \\ -S_1 & C_1 & 0 & 72C_1S_1C_{23} + 72C_1S_1C_2 - 72S_1C_1C_{23} - 72S_1C_2C_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^{3}A_0 = \begin{bmatrix} C_1C_{23} & S_1C_{23} & -S_{23} & -72C_{23}^2 - 72C_2C_{23} - 72S_{23}^2 - 72S_2S_{23} + 11.25S_{23} \\ -C_1S_{23} & -S_1S_{23} & -C_{23} & 72C_{23}S_{23} + 72C_2S_{23} - 72S_{23}C_{23} - 72S_2C_{23} + 11.25C_{23} \\ -S_1 & C_1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^{3}A_0 = \begin{bmatrix} C_1C_{23} & S_1C_{23} & -S_{23} & -72 - 72C_3 + 11.25S_{23} \\ -C_1S_{23} & -S_1S_{23} & -C_{23} & 72S_3 + 11.25C_{23} \\ -S_1 & C_1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

Column 1

$$ {}^{3}P_{0/3} \times {}^{3}z_0 = \begin{bmatrix} -72 - 72C_3 + 11.25S_{23} \\ 72S_3 + 11.25C_{23} \\ 0 \end{bmatrix} \times \begin{bmatrix} -S_{23} \\ -C_{23} \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 72C_{23} + 72C_{23}C_3 - 11.25S_{23}C_{23} + 72S_3S_{23} + 11.25C_{23}S_{23} \end{bmatrix} $$

$$ {}^{3}P_{0/3} \times {}^{3}z_0 \quad \begin{bmatrix} 0 \\ 0 \\ 72C_{23} + 72C_2 \end{bmatrix} $$

Column 2

$$ {}^{3}P_{1/3} \times {}^{3}z_1 = \begin{bmatrix} -72 - 72C_3 \\ 72S_3 \\ 0 \end{bmatrix} \times \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 72S_3 \\ 72 + 72C_3 \\ 0 \end{bmatrix} $$

Column 3

$$ {}^{3}P_{2/3} \times {}^{3}z_2 = \begin{bmatrix} -72 \\ 0 \\ 0 \end{bmatrix} \times \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 72 \\ 0 \end{bmatrix} $$

$$ J_3 = \begin{bmatrix} 0 & 72S_3 & 0 & 0 \\ 0 & 72 + 72C_3 & 72 & 0 \\ 72C_{23} + 72C_2 & 0 & 0 & 0 \\ -S_{23} & 0 & 0 & 0 \\ -C_{23} & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \end{bmatrix} $$

Calculate W_3 $\qquad W_3 = J_3^T V_3 J_3$

$$W_3 = J_3^T \begin{bmatrix} m_3 & 0 & 0 & 0 & 0 & 0 \\ 0 & m_3 & 0 & 0 & 0 & C_x m_3 \\ 0 & 0 & m_3 & 0 & -C_x m_3 & 0 \\ 0 & 0 & 0 & I_{xx} & 0 & 0 \\ 0 & 0 & -C_x m_3 & 0 & I_{yy} & 0 \\ 0 & C_x m_3 & 0 & 0 & 0 & I_{zz} \end{bmatrix} \begin{bmatrix} 0 & l_2 S_3 & 0 & 0 \\ 0 & l_2 + l_2 C_3 & l_2 & 0 \\ l_2 C_{23} + l_2 C_2 & 0 & 0 & 0 \\ -S_{23} & 0 & 0 & 0 \\ -C_{23} & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \end{bmatrix}$$

$$W_3 = \begin{bmatrix} 0 & 0 & l_2 C_{23} + l_2 C_2 & -S_{23} & -C_{23} & 0 \\ l_2 S_3 & l_2 + l_2 C_3 & 0 & 0 & 0 & 1 \\ 0 & l_2 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & (l_2 S_3 m_3) & 0 & 0 \\ 0 & (l_2 m_3 + l_2 C_3 m_3 + C_x m_3) & (l_2 m_3 + C_x m_3) & 0 \\ (l_2 C_{23} m_3 + l_2 C_2 m_3 + C_{23} C_x m_3) & 0 & 0 & 0 \\ (-I_{xx} S_{23}) & 0 & 0 & 0 \\ (-l_2 C_{23} C_x m_3 - l_2 C_2 C_x m_3 - I_{yy} C_{23}) & 0 & 0 & 0 \\ 0 & (l_2 C_x m_3 + l_2 C_3 C_x m_3 + I_{zz}) & (l_2 C_x m_3 + I_{zz}) & 0 \end{bmatrix}$$

$$W_3 = \begin{bmatrix} (l_2^2 C_{23}^2 m_3 + l_2^2 C_{23} C_2 m_3 + l_2 C_{23}^2 C_x m_3 \\ l_2^2 C_2 C_{23} m_3 + l_2^2 C_2^2 m_3 + l_2 C_2 C_{23} C_x m_3 \\ + I_{xx} S_{23}^2 + l_2 C_{23}^2 C_x m_3 \\ + l_2 C_{23} C_2 C_x m_3 + I_{yy} C_{23}^2) & 0 & 0 & 0 \\ 0 & \begin{pmatrix} l_2^2 S_3^2 m_3 + l_2^2 m_3 + l_2^2 C_3 m_3 \\ + l_2 C_x m_3 + l_2^2 C_3 m_3 + l_2^2 C_3^2 m_3 \\ + l_2 C_3 C_x m_3 + l_2 C_x m_3 \\ + l_2 C_3 C_x m_3 + I_{zz} \end{pmatrix} \begin{pmatrix} l_2^2 m_3 + l_2 C_x m_3 + l_2^2 C_3 m_3 \\ + l_2 C_3 C_x m_3 + l_2 C_x m_3 + I_{zz} \end{pmatrix} & 0 \\ 0 & \begin{pmatrix} l_2^2 m_3 + l_2^2 C_3 m_3 + l_2 C_x m_3 \\ + l_2 C_x m_3 + l_2 C_3 C_x m_3 + I_{zz} \end{pmatrix} \begin{pmatrix} l_2^2 m_3 + l_2 C_x m_3 \\ + l_2 C_x m_3 + I_{zz} \end{pmatrix} & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$W_3 = \begin{bmatrix} \begin{pmatrix} r_2^2 C_{23}^2 m_3 + (2)(r_2)^2 (C_{23} C_2 m_3 \\ + r_2^2 C_2^2 m_3 + (2)(r_2)(C_{23}^2 C_x m_3 \\ + (2)(r_2)(C_{23} C_2 C_x m_3 \\ + I_{xx} S_{23}^2 + I_{yy} C_{23}^2 \end{pmatrix} & 0 & 0 & 0 \\ 0 & \begin{pmatrix} (2)(r_2)^2 m_3 + (2)(r_2)^2 C_3 m_3 \\ + (2)(r_2) C_x m_3 + I_{zz} \\ + (2)(r_2) C_3 C_x m_3 \end{pmatrix} & \begin{pmatrix} r_2^2 m_3 + r_2^2 C_3 m_3 \\ + (2)(r_2) C_x m_3 + r_2 C_3 C_x m_3 \\ + I_{zz} \end{pmatrix} & 0 \\ 0 & \begin{pmatrix} r_2^2 m_3 + r_2^2 C_3 m_3 \\ + (2)(r_2) C_x m_3 + r_2 C_3 C_x m_3 \\ + I_{zz} \end{pmatrix} & \begin{pmatrix} r_2^2 m_3 + (2)(r_2) C_x m_3 + I_{zz} \end{pmatrix} & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Calculate $\frac{\partial W_3}{\partial q_1}$, $\frac{\partial W_3}{\partial q_2}$, $\frac{\partial W_3}{\partial q_3}$, $\frac{\partial W_3}{\partial q_4}$

$$\frac{\partial W_3}{\partial q_1} = \frac{\partial W_3}{\partial q_4} = [0]$$

$$\frac{\partial W_3}{\partial q_2} = \begin{bmatrix} \begin{pmatrix} -(2)(72)^2 C_{23} S_{23} M_3 - (2)(72)^2 S_{23} C_2 M_3 - (2)(72)^2 C_{23} S_2 M_3 \\ -(2)(72)^2 C_2 S_2 M_3 - (4)(72) C_{23} S_{23} C_x M_3 \\ -(2)(72) S_{23} C_2 C_x M_3 - (2)(72) C_{23} S_2 C_x M_3 \\ + (2) I_{xx} S_{23} C_{23} - (2) I_{yy} C_{23} S_{23} \end{pmatrix} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\frac{\partial W_3}{\partial q_3} = \begin{bmatrix} \begin{pmatrix} -(2)(72)^2 C_{23} S_{23} M_3 - (2)(72)^2 S_{23} C_2 M_3 \\ -(4)(72) C_{23} S_{23} C_x M_3 \\ -(2)(72) S_{23} C_2 C_x M_3 \\ + (2) I_{xy} S_{23} C_{23} - (2) I_{yy} C_{23} S_{23} \end{pmatrix} & 0 & 0 & 0 \\ 0 & \begin{pmatrix} -(2)(72)^2 S_3 M_3 \\ -(2)(72) S_3 C_x M_3 \end{pmatrix} & \begin{pmatrix} -(72)^2 S_3 M_3 \\ -(72) S_3 C_x M_3 \end{pmatrix} & 0 \\ 0 & \begin{pmatrix} -(72)^2 S_3 M_3 \\ -(72) S_3 C_x M_3 \end{pmatrix} & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\frac{\partial W_3}{\partial q_3} = \begin{bmatrix} \frac{\partial W_3}{\partial q_3}(1,1) & 0 & 0 & 0 \\ 0 & \frac{\partial W_3}{\partial q_3}(2,2) & \frac{\partial W_3}{\partial q_3}(2,3) & 0 \\ 0 & \frac{\partial W_3}{\partial q_3}(3,2) & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Calculate I_4:

$$ {}^3A_4 = \begin{bmatrix} C_4 & -S_4 & 0 & 36C_4 \\ S_4 & C_4 & 0 & 36S_4 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^4A_3 = \begin{bmatrix} C_4 & S_4 & 0 & -36C_4^2 - 36S_4^2 \\ -S_4 & C_4 & 0 & 36C_4S_4 - 36S_4C_4 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} C_4 & S_4 & 0 & -36 \\ -S_4 & C_4 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^2A_4 = {}^2A_3\,{}^3A_4 = \begin{bmatrix} C_3 & -S_3 & 0 & 72C_3 \\ S_3 & C_3 & 0 & 72S_3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} C_4 & -S_4 & 0 & 36C_4 \\ S_4 & C_4 & 0 & 36S_4 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^2A_4 = \begin{bmatrix} C_3C_4 - S_3S_4 & -C_3S_4 - S_3C_4 & 0 & 36C_3C_4 - 36S_3S_4 + 72C_3 \\ S_3C_4 + C_3S_4 & -S_3S_4 + C_3C_4 & 0 & 36S_3C_4 + 36C_3S_4 + 72S_3 \\ 0 & 0 & 1 & \\ 0 & 0 & 0 & \end{bmatrix} $$

$$ {}^2A_4 = \begin{bmatrix} C_{34} & -S_{34} & 0 & 36C_{34} + 72C_3 \\ S_{34} & C_{34} & 0 & 36S_{34} + 72S_3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^4A_2 = \begin{bmatrix} C_{34} & S_{34} & 0 & -36C_{34}^2 - 72C_3C_{34} - 36S_{34}^2 - 72S_3S_{34} \\ -S_{34} & C_{34} & 0 & 36C_{34}S_{34} + 72C_3S_{34} - 36S_{34}C_{34} - 72S_3C_{34} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^{4}A_{2} = \begin{bmatrix} C_{34} & S_{34} & 0 & -36 - 72C_4 \\ -S_{34} & C_{34} & 0 & 72S_4 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^{1}A_{4} = {}^{1}A_{2}\,{}^{2}A_{4} = \begin{bmatrix} C_2 & -S_2 & 0 & 72C_2 \\ S_2 & C_2 & 0 & 72S_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} C_{34} & -S_{34} & 0 & 36C_{34} + 72C_3 \\ S_{34} & C_{34} & 0 & 36S_{34} + 72S_3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^{1}A_{4} = \begin{bmatrix} C_2C_{34} - S_2S_{34} & -C_2S_{34} - S_2C_{34} & 0 & 36C_2C_{34} + 72C_2C_3 - 36S_2S_{34} - 72S_2S_3 + 72C_2 \\ S_2C_{34} + C_2S_{34} & -S_2S_{34} + C_2C_{34} & 0 & 36S_2C_{34} + 72S_2C_3 + 36C_2S_{34} + 72C_2S_3 + 72S_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^{1}A_{4} = \begin{bmatrix} C_{234} & -S_{234} & 0 & 36(C_2C_{34} - S_2S_{34}) + 72(C_2C_3 - S_2S_3) + 72C_2 \\ S_{234} & C_{234} & 0 & 36(S_2C_{34} + C_2S_{34}) + 72(S_2C_3 + C_2S_3) + 72S_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^{1}A_{4} = \begin{bmatrix} C_{234} & -S_{234} & 0 & 36C_{234} + 72C_{23} + 72C_2 \\ S_{234} & C_{234} & 0 & 36S_{234} + 72S_{23} + 72S_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^{4}A_{1} = \begin{bmatrix} C_{234} & S_{234} & 0 & \begin{pmatrix} -36C_{234}^2 - 72C_{23}C_{234} - 72C_2C_{234} \\ -36S_{234}^2 - 72S_{23}S_{234} - 72S_2S_{234} \end{pmatrix} \\ -S_{234} & C_{234} & 0 & \begin{pmatrix} 36C_{234}S_{234} + 72C_{23}S_{234} + 72C_2S_{234} \\ -36S_{234}C_{234} - 72S_{23}C_{234} - 72S_2C_{234} \end{pmatrix} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$
{}^{4}A_{1} = \begin{bmatrix} C_{234} & S_{234} & 0 & -36 - 72C_4 - 72C_{34} \\ -S_{234} & C_{234} & 0 & 72S_4 + 72S_{34} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

$$
{}^{0}A_{4} = {}^{0}A_{1}\,{}^{1}A_{4} = \begin{bmatrix} C_1 & 0 & -S_1 & 0 \\ S_1 & 0 & C_1 & 0 \\ 0 & -1 & 0 & 11.25 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} C_{234} & -S_{234} & 0 & 36C_{234} + 72C_{23} + 72C_2 \\ S_{234} & C_{234} & 0 & 36S_{234} + 72S_{23} + 72S_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

$$
{}^{0}A_{4} = \begin{bmatrix} C_1C_{234} & -C_1S_{234} & -S_1 & 36C_1C_{234} + 72C_1C_{23} + 72C_1C_2 \\ S_1C_{234} & -S_1S_{234} & C_1 & 36S_1C_{234} + 72S_1C_{23} + 72S_1C_2 \\ -S_{234} & -C_{234} & 0 & -36S_{234} - 72S_{23} - 72S_2 + 11.25 \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

$$
{}^{4}A_{0} = {}^{4}A_{1}\,{}^{1}A_{0} = \begin{bmatrix} C_{234} & S_{234} & 0 & -36 - 72C_4 - 72C_{34} \\ -S_{234} & C_{234} & 0 & 72S_4 + 72S_{34} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} C_1 & S_1 & 0 & 0 \\ 0 & 0 & -1 & 11.25 \\ -S_1 & C_1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

$$
{}^{4}A_{0} = \begin{bmatrix} C_1C_{234} & S_1C_{234} & -S_{234} & -36 - 72C_4 - 72C_{34} + 11.25S_{234} \\ -C_1S_{234} & -S_1S_{234} & -C_{234} & 72S_4 + 72S_{34} + 11.25C_{234} \\ -S_1 & C_1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

Column 1

$$ {}^4P_{0/4} \times {}^4Z_0 = \begin{bmatrix} -36 - 72C_4 - 72C_{34} + 11.25S_{234} \\ 72S_4 + 72S_{34} + 11.25C_{234} \\ 0 \end{bmatrix} \times \begin{bmatrix} -S_{234} \\ -C_{234} \\ 0 \end{bmatrix} $$

$$ {}^4P_{0/4} \times {}^4Z_0 \begin{bmatrix} 0 \\ 0 \\ +36C_{234} + 72C_4C_{234} + 72C_{34}C_{234} - 11.25S_{234}C_{234} + 72S_4S_{234} + 72S_{34}S_{234} + 11.25C_{234}S_{234} \end{bmatrix} $$

$$ {}^4P_{0/4} \times {}^4Z_0 = \begin{bmatrix} 0 \\ 0 \\ 36C_{234} + 72C_{23} + 72C_2 \end{bmatrix} $$

Column 2

$$ {}^4P_{1/4} \times {}^4Z_1 = \begin{bmatrix} -36 - 72C_4 - 72C_{34} \\ 72S_4 + 72S_{34} \\ 0 \end{bmatrix} \times \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 72S_4 + 72S_{34} \\ 36 + 72C_4 + 72C_{34} \\ 0 \end{bmatrix} $$

Column 3

$$ {}^4P_{2/4} \times {}^4Z_2 = \begin{bmatrix} -36 - 72C_4 \\ 72S_4 \\ 0 \end{bmatrix} \times \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 72S_4 \\ 36 + 72C_4 \\ 0 \end{bmatrix} $$

Column 4

$$ {}^4P_{3/4} \times {}^4Z_3 = \begin{bmatrix} -36 \\ 0 \\ 0 \end{bmatrix} \times \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 36 \\ 0 \end{bmatrix} $$

$$J_4 = \begin{bmatrix} 0 & 72s_4 + 72s_{34} & 72s_4 & 0 \\ 0 & 36 + 72c_4 + 72c_{34} & 36 + 72c_4 & 36 \\ 36c_{234} + 72c_{23} + 72c_2 & 0 & 0 & 0 \\ -s_{234} & 0 & 0 & 0 \\ -c_{234} & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \end{bmatrix}$$

Calculate W_4 $\qquad W_4 = J_4^T U_4 J_4$

$$W_4 = J_4^T \begin{bmatrix} m_4 & 0 & 0 & 0 & 0 & 0 \\ 0 & m_4 & 0 & 0 & 0 & C_x m_4 \\ 0 & 0 & m_4 & 0 & -C_x m_4 & 0 \\ 0 & 0 & 0 & I_{xx} & 0 & 0 \\ 0 & 0 & -C_x m_4 & 0 & I_{yy} & 0 \\ 0 & C_x m_4 & 0 & 0 & 0 & I_{zz} \end{bmatrix} \begin{bmatrix} 0 & 72S_4 + 72S_{34} & 72S_4 & 0 \\ 0 & 36 + 72C_4 + 72C_{34} & 36 + 72C_4 & 36 \\ 36C_{234} + 72C_{23} + 72C_2 & 0 & 0 & 0 \\ -S_{234} & 0 & 0 & 0 \\ -C_{234} & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \end{bmatrix}$$

$$W_4 = \begin{bmatrix} 0 & 0 & 36C_{234} + 72C_{23} + 72C_2 & -S_{234} & -C_{234} & 0 \\ 72S_4 + 72S_{34} & 36 + 72C_4 + 72C_{34} & 0 & 0 & 0 & 1 \\ 72S_4 & 36 + 72C_4 & 0 & 0 & 0 & 1 \\ 0 & 36 & 0 & 0 & 0 & 1 \end{bmatrix} *$$

$$\begin{bmatrix} 0 & 72S_4 m_4 + 72S_{34} m_4 & 72S_4 m_4 & 0 \\ 0 & \left(36m_4 + 72C_4 m_4 + 72C_{34} m_4 + C_x m_4\right) & \left(36m_4 + 72C_4 m_4 + C_x m_4\right) & 36m_4 + C_x m_4 \\ \left(36C_{234} m_4 + 72C_{23} m_4 + 72C_2 m_4 + C_{234} C_x m_4\right) & 0 & 0 & 0 \\ -I_{xx} S_{234} & 0 & 0 & 0 \\ \left(-36C_{234} C_x m_4 - 72C_{23} C_x m_4 - 72C_2 C_x m_4 - I_{yy} C_{234}\right) & 0 & 0 & 0 \\ 0 & \left(36C_x m_4 + 72C_4 C_x m_4 + 72C_{34} C_x m_4 + I_{zz}\right) & \left(36C_x m_4 + 72C_4 C_x m_4 + I_{zz}\right) & 36C_x m_4 + I_{zz} \end{bmatrix}$$

$$W_4 = \begin{bmatrix} 36^2 C_{234}^2 m_4 + (36)(72) C_{23} C_{234} m_4 + (36)(72) C_2 C_{234} m_4 + (36)(72) C_{234} C_{23} m_4 + 72^2 C_{23}^2 m_4 + 72^2 C_2 C_{23} m_4 + (36)(72) C_{234} C_2 m_4 + 72^2 C_{23} C_2 m_4 + 72^2 C_2^2 m_4 + 36 C_{234}^2 C_x m_4 + 72 C_{23} C_{234} C_x m_4 + 72 C_2 C_{234} C_x m_4 + I_{xx} S_{234}^2 + 36 C_{234}^2 C_x m_4 + 72 C_{234} C_{23} C_x m_4 + 72 C_{234} C_2 C_x m_4 + I_{yy} C_{234}^2 & 0 & 0 & 0 \\ 0 & \begin{pmatrix} 72^2 S_4^2 m_4 + 72^2 S_4 S_{34} m_4 + 72^2 S_4 S_{34} m_4 + 72^2 S_{34}^2 m_4 + 36^2 m_4 + (36)(72) C_4 m_4 + (36)(72) C_{34} m_4 + 36 C_x m_4 + (36)(72) C_4 m_4 + 72^2 C_4^2 m_4 + 72^2 C_4 C_{34} m_4 + 72 C_4 C_x m_4 + (36)(72) C_{34} m_4 + 72^2 C_{34} C_4 m_4 + 72^2 C_{34}^2 m_4 + 72 C_{34} C_x m_4 + 36 C_x m_4 + 72 C_4 C_x m_4 + 72 C_{34} C_x m_4 + I_{zz} \end{pmatrix} & \begin{pmatrix} 72^2 S_4^2 m_4 + 72^2 S_4 S_{34} m_4 + 36^2 m_4 + (36)(72) C_4 m_4 + 36 C_x m_4 + (36)(72) C_4 m_4 + 72^2 C_4^2 m_4 + 72 C_4 C_x m_4 + (36)(72) C_{34} m_4 + 72^2 C_4 C_{34} m_4 + 72 C_{34} C_x m_4 + 36 C_x m_4 + 72 C_4 C_x m_4 + I_{zz} \end{pmatrix} & \begin{pmatrix} 36^2 m_4 + 36 C_x m_4 + (36)(72) C_4 m_4 + 72 C_4 C_x m_4 + (36)(72) C_{34} m_4 + 72 C_{34} C_x m_4 + 36 C_x m_4 + I_{zz} \end{pmatrix} \\ 0 & \begin{pmatrix} 72^2 S_4^2 m_4 + 72^2 S_4 S_{34} m_4 + 36^2 m_4 + (36)(72) C_4 m_4 + (36)(72) C_{34} m_4 + 36 C_x m_4 + (36)(72) C_4 m_4 + 72^2 C_4^2 m_4 + 72^2 C_4 C_{34} m_4 + 36 C_x m_4 + 72 C_4 C_x m_4 + 72 C_{34} C_x m_4 + I_{zz} \end{pmatrix} & \begin{pmatrix} 72^2 S_4^2 m_4 + 36^2 m_4 + (36)(72) C_4 m_4 + 36 C_x m_4 + (36)(72) C_4 m_4 + 72^2 C_4^2 m_4 + 72 C_4 C_x m_4 + 36 C_x m_4 + 72 C_4 C_x m_4 + I_{zz} \end{pmatrix} & \begin{pmatrix} 36^2 m_4 + 36 C_x m_4 + (36)(72) C_4 m_4 + 72 C_4 C_x m_4 + 36 C_x m_4 + I_{zz} \end{pmatrix} \\ 0 & \begin{pmatrix} 36^2 m_4 + (36)(72) C_4 m_4 + (36)(72) C_{34} m_4 + 36 C_x m_4 + 36 C_x m_4 + 72 C_4 C_x m_4 + 72 C_{34} C_x m_4 + I_{zz} \end{pmatrix} & \begin{pmatrix} 36^2 m_4 + (36)(72) C_4 m_4 + 36 C_x m_4 + 36 C_x m_4 + 72 C_4 C_x m_4 + I_{zz} \end{pmatrix} & \begin{pmatrix} 36^2 m_4 + 36 C_x m_4 + 36 C_x m_4 + I_{zz} \end{pmatrix} \end{bmatrix}$$

$$W_4 = \begin{bmatrix} \left(\begin{array}{l} 36^2 C_{234}^2 m_4 + (2)(36)(72) C_{23} C_{234} m_4 \\ + (2)(36)(72) C_2 C_{234} m_4 + 72^2 C_{23}^2 m_4 \\ + (2)(72)^2 C_2 C_{23} m_4 + 72^2 C_2^2 m_4 \\ + (2)(36) C_{234}^2 C_x m_4 \\ + (2)(72) C_{23} C_{234} C_x m_4 \\ + (2)(72) C_2 C_{234} C_x m_4 \\ + I_{xx} S_{234}^2 + I_{yy} C_{234}^2 \end{array}\right) & \\ 0 & \left(\begin{array}{l} (2)72^2 m_4 + (2)(72)^2 S_4 S_{34} m_4 \\ + 36^2 m_4 + (2)(36)(72) C_4 m_4 \\ + (2)(36)(72) C_{34} m_4 + (2)(36) C_x m_4 \\ + (2)(72)^2 C_{34} C_4 m_4 \\ + (2)(72) C_4 C_x m_4 \\ + (2)(72) C_{34} C_x m_4 + I_{zz} \end{array}\right) \left(\begin{array}{l} 72^2 m_4 + 36^2 m_4 \\ + 72^2 S_4 S_{34} m_4 \\ + 72^2 C_4 C_{34} m_4 \\ + (2)(36)(72) C_4 m_4 \\ + (2)(36) C_x m_4 + I_{zz} \\ + (2)(72) C_4 C_x m_4 \\ + (36)(72) C_{34} m_4 + 72 C_{34} C_x m_4 \end{array}\right) \left(\begin{array}{l} 36^2 m_4 + (2)(36) C_x m_4 \\ + (36)(72) C_4 m_4 \\ + (36)(72) C_{34} m_4 \\ + 72 C_4 C_x m_4 \\ + 72 C_{34} C_x m_4 \\ + I_{zz} \end{array}\right) \\ 0 & \left(\begin{array}{l} 72^2 m_4 + 36^2 m_4 \\ + 72^2 S_4 S_{34} m_4 + 72^2 C_4 C_{34} m_4 \\ + (2)(36)(72) C_4 m_4 + (2)(36) C_x m_4 \\ + (36)(72) C_{34} m_4 + 72 C_{34} C_x m_4 \\ + (2)(72) C_4 C_x m_4 + I_{zz} \end{array}\right) \left(\begin{array}{l} 72^2 m_4 + 36^2 m_4 \\ + (2)(36)(72) C_4 m_4 \\ + (2)(36) C_x m_4 + I_{zz} \\ + (2)(72) C_4 C_x m_4 \end{array}\right) \left(\begin{array}{l} 36^2 m_4 + (2)(36) C_x m_4 \\ + (36)(72) C_4 m_4 \\ + (72) C_4 C_x m_4 \\ + I_{zz} \end{array}\right) \\ 0 & \left(\begin{array}{l} 36^2 m_4 + (36)(72) C_4 m_4 \\ + (36)(72) C_{34} m_4 + (2)(36) C_x m_4 \\ + 72 C_4 C_x m_4 \\ + 72 C_{34} C_x m_4 + I_{zz} \end{array}\right) \left(\begin{array}{l} 36^2 m_4 + (36)(72) C_4 m_4 \\ + (2)(36) C_x m_4 \\ + 72 C_4 C_x m_4 + I_{zz} \end{array}\right) \left(\begin{array}{l} 36^2 m_4 \\ + (2)(36) C_x m_4 \\ + I_{zz} \end{array}\right) \end{bmatrix}$$

Calculate $\frac{\partial W_4}{\partial q_1}, \frac{\partial W_4}{\partial q_2}, \frac{\partial W_4}{\partial q_3}, \frac{\partial W_4}{\partial q_4}$

$$\frac{\partial W_4}{\partial q_1} = [0]$$

$$\frac{\partial W_4}{\partial q_2} = \begin{bmatrix} -(2)(36)^2 C_{234} S_{234} m_4 - (2)(36)(72) S_{23} C_{234} m_4 - (2)(36)(72) C_{23} S_{234} m_4 & & & \\ -(2)(36)(72) S_2 C_{234} m_4 - (2)(36)(72) C_2 S_{234} m_4 - (2)(72)^2 C_{23} S_{23} m_4 & & & \\ -(2)(72)^2 S_2 C_{23} m_4 - (2)(72)^2 C_2 S_{23} m_4 - (2)(72)^2 C_2 S_2 m_4 & 0 & 0 & 0 \\ -(4)(36) C_{234} S_{234} C_x m_4 - (2)(72) S_{23} C_{234} C_x m_4 & & & \\ -(2)(72) C_{23} S_{234} C_x m_4 - (2)(72) S_2 C_{234} C_x m_4 & & & \\ -(2)(72) C_2 S_{234} C_x m_4 + (2) I_{xx} S_{234} C_{234} & & & \\ \quad -(2) I_{yy} C_{234} S_{234} & & & \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\frac{\partial w_4}{\partial q_3} = \begin{bmatrix} \begin{pmatrix} -(2)(36)^2 C_{234} S_{234} m_4 - (2)(36)(72) S_{23} C_{234} m_4 \\ -(2)(36)(72) C_{23} S_{234} m_4 - (2)(36)(72) C_2 S_{234} m_4 \\ -(2)(72)^2 C_{23} S_{23} m_4 - (2)(72)^2 C_2 S_{23} m_4 \\ -(4)(36) C_{234} S_{234} C_x m_4 \\ -(2)(72) S_{23} C_{234} C_x m_4 \\ -(2)(72) C_{23} S_{234} C_x m_4 \\ -(2)(72) C_2 S_{234} C_x m_4 \\ +(2) I_{xx} S_{234} C_{234} - (2) I_{yy} C_{234} S_{234} \end{pmatrix} & 0 & 0 & 0 \\ 0 & \begin{pmatrix} +(2)(72)^2 S_4 C_{34} m_4 \\ -(2)(36)(72) S_{34} m_4 \\ -(2)(72)^2 S_{34} C_4 m_4 \\ -(2)(72) S_{34} C_x m_4 \end{pmatrix} & \begin{pmatrix} +72^2 S_4 C_{34} m_4 \\ -72^2 C_4 S_{34} m_4 \\ -(36)(72) S_{34} m_4 \\ -72 S_{34} C_x m_4 \end{pmatrix} & \begin{pmatrix} -(36)(72) S_{34} m_4 \\ -72 S_{34} C_x m_4 \end{pmatrix} \\ 0 & \begin{pmatrix} +72^2 S_4 C_{34} m_4 \\ -72^2 C_4 S_{34} m_4 \\ -(36)(72) S_{34} m_4 \\ -(72) S_{34} C_x m_4 \end{pmatrix} & 0 & 0 \\ 0 & \begin{pmatrix} -(36)(72) S_{34} m_4 \\ -(72) S_{34} C_x m_4 \end{pmatrix} & 0 & 0 \end{bmatrix}$$

$$\frac{\partial w_4}{\partial q_3} = \begin{bmatrix} \frac{\partial w_4}{\partial q_3}(1,1) & 0 & 0 & 0 \\ 0 & \frac{\partial w_4}{\partial q_3}(2,2) & \frac{\partial w_4}{\partial q_3}(2,3) & \frac{\partial w_4}{\partial q_3}(2,4) \\ 0 & \frac{\partial w_4}{\partial q_3}(3,2) & 0 & 0 \\ 0 & \frac{\partial w_4}{\partial q_3}(4,2) & 0 & 0 \end{bmatrix}$$

$$\frac{\partial W_4}{\partial q_4} =$$

$\frac{\partial W_4}{\partial q_4}(1,1)$

$$\begin{pmatrix} -(2)(36)^2 C_{234} S_{234} M_4 \\ -(2)(36)(72) C_{23} S_{234} M_4 \\ -(2)(36)(72) C_2 S_{234} M_4 \\ -(4)(36) C_{234} S_{234} C_x M_4 \\ -(2)(72) C_{23} S_{234} C_x M_4 \\ -(2)(72) C_2 S_{234} C_x M_4 \\ +(2) I_{xx} S_{234} C_{234} \\ -(2) I_{yy} C_{234} S_{234} \end{pmatrix}$$

0 0 0

$\frac{\partial W_4}{\partial q_4}(2,2)$

$$\begin{pmatrix} (2)(72)^2 C_4 S_{34} M_4 \\ +(2)(72)^2 S_4 C_{34} M_4 \\ -(2)(36)(72) S_4 M_4 \\ -(2)(36)(72) S_{34} M_4 \\ -(2)(72)^2 S_{34} C_4 M_4 \\ -(2)(72)^2 C_{34} S_4 M_4 \\ -(2)(72) S_4 C_x M_4 \\ -(2)(72) S_{34} C_x M_4 \end{pmatrix}$$

$\frac{\partial W_4}{\partial q_4}(2,3)$

$$\begin{pmatrix} 72^2 C_4 S_{34} M_4 \\ +72^2 S_4 C_{34} M_4 \\ -72^2 S_4 C_{34} M_4 \\ -72^2 C_4 S_{34} M_4 \\ -(2)(36)(72) S_4 M_4 \\ -(2)(72) S_4 C_x M_4 \\ -(36)(72) S_{34} M_4 \\ -72 S_{34} C_x M_4 \end{pmatrix}$$

$\frac{\partial W_4}{\partial q_4}(2,4)$

$$\begin{pmatrix} -(36)(72) S_4 M_4 \\ -(36)(72) S_{34} M_4 \\ -(72) S_4 C_x M_4 \\ -(72) S_{34} C_x M_4 \end{pmatrix}$$

0

$\frac{\partial W_4}{\partial q_4}(3,2)$

$$\begin{pmatrix} +(72)^2 C_4 S_{34} M_4 \\ +(72)^2 S_4 C_{34} M_4 \\ -(72)^2 S_4 C_{34} M_4 \\ -(72)^2 C_4 S_{34} M_4 \\ -(2)(36)(72) S_4 M_4 \\ -(36)(72) S_{34} M_4 \\ -(72) S_{34} C_x M_4 \\ -(2)(72) S_4 C_x M_4 \end{pmatrix}$$

$\frac{\partial W_4}{\partial q_4}(3,3)$

$$\begin{pmatrix} -(2)(36)(72) S_4 M_4 \\ -(2)(72) S_4 C_x M_4 \end{pmatrix}$$

$\frac{\partial W_4}{\partial q_4}(3,4)$

$$\begin{pmatrix} -(36)(72) S_4 M_4 \\ -(72) S_4 C_x M_4 \end{pmatrix}$$

0

$\frac{\partial W_4}{\partial q_4}(4,2)$

$$\begin{pmatrix} -(36)(72) S_4 M_4 \\ -(36)(72) S_{34} M_4 \\ -(72) S_4 C_x M_4 \\ -72 S_{34} C_x M_4 \end{pmatrix}$$

$\frac{\partial W_4}{\partial q_4}(4,3)$

$$\begin{pmatrix} -(36)(72) S_4 M_4 \\ -(72) S_4 C_x M_4 \end{pmatrix}$$

0

Calculate $\tilde{\omega}$

$$\tilde{\omega} = \begin{bmatrix} \dot{\underline{\epsilon}}^T \frac{\partial \underline{w}}{\partial q_1} \\ \vdots \\ \dot{\underline{\epsilon}}^T \frac{\partial \underline{w}}{\partial q_4} \end{bmatrix} = \begin{bmatrix} [\dot{q}_1\ \dot{q}_2\ \dot{q}_3\ \dot{q}_4] \frac{\partial \underline{w}}{\partial q_1} \\ \vdots \\ [\dot{q}_1\ \dot{q}_2\ \dot{q}_3\ \dot{q}_4] \frac{\partial \underline{w}}{\partial q_4} \end{bmatrix}$$

$$\tilde{\omega} = \left[\begin{array}{c|c|c|c} 0 & 0 & 0 & 0 \\ \hline \dot{q}_1\left[\frac{\partial w_2}{\partial q_2}(1,1) + \frac{\partial w_3}{\partial q_2}(1,1) + \frac{\partial w_4}{\partial q_2}(1,1)\right] & 0 & 0 & 0 \\ \hline \dot{q}_1\left[\frac{\partial w_3}{\partial q_3}(1,1) + \frac{\partial w_4}{\partial q_3}(1,1)\right] & \begin{array}{l}\dot{q}_2\left[\frac{\partial w_3}{\partial q_3}(2,2) + \frac{\partial w_4}{\partial q_3}(2,2)\right] \\ +\dot{q}_3\left[\frac{\partial w_3}{\partial q_3}(3,2) + \frac{\partial w_4}{\partial q_3}(3,2)\right] \\ +\dot{q}_4\left[\frac{\partial w_4}{\partial q_3}(4,2)\right]\end{array} & \dot{q}_2\left[\frac{\partial w_3}{\partial q_3}(2,3) + \frac{\partial w_4}{\partial q_3}(2,3)\right] & \dot{q}_2\left[\frac{\partial w_4}{\partial q_3}(2,4)\right] \\ \hline \dot{q}_1\left[\frac{\partial w_4}{\partial q_4}(1,1)\right] & \begin{array}{l}\dot{q}_2\left[\frac{\partial w_4}{\partial q_4}(2,2)\right] \\ \dot{q}_3\left[\frac{\partial w_4}{\partial q_4}(3,2)\right] \\ \dot{q}_4\left[\frac{\partial w_4}{\partial q_4}(4,2)\right]\end{array} & \begin{array}{l}\dot{q}_2\left[\frac{\partial w_4}{\partial q_4}(2,3)\right] \\ +\dot{q}_3\left[\frac{\partial w_4}{\partial q_4}(3,3)\right] \\ +\dot{q}_4\left[\frac{\partial w_4}{\partial q_4}(4,3)\right]\end{array} & \begin{array}{l}\dot{q}_2\left[\frac{\partial w_4}{\partial q_4}(2,4)\right] \\ +\dot{q}_3\left[\frac{\partial w_4}{\partial q_4}(3,4)\right]\end{array} \end{array}\right]$$

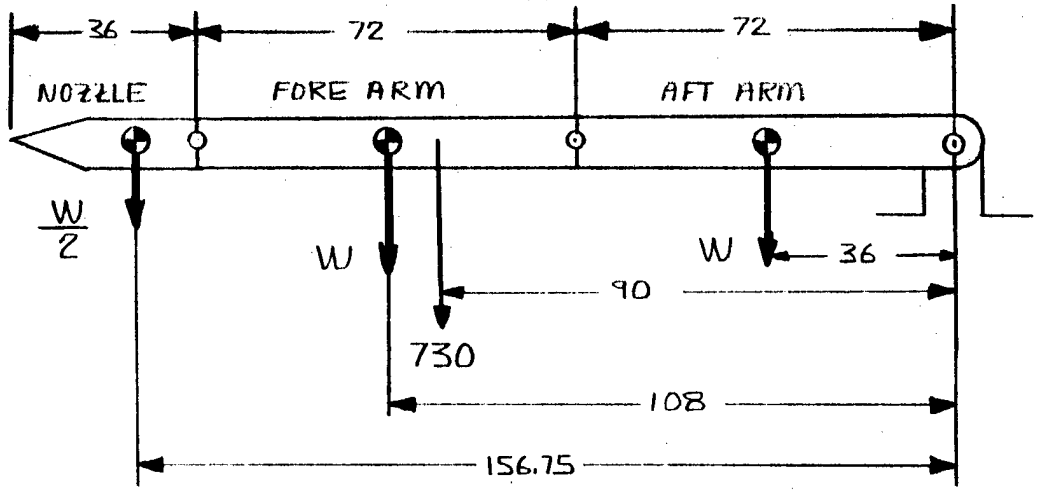
APPENDIX B

MASS AND INERTIA APPROXIMATIONS

WEIGHT OF EACH LINK

ACTUAL WEIGHTS ARE UNKNOWN.

MANUFACTUER REPORTED THAT THE TORQUE ABOUT THE SHOULDER JOINT IS EQUIVALENT TO 730 lbs AT A DISTANCE OF 90" WHEN THE ARM IS FULLY EXTENDED AND FILLED WITH FUEL



$$\frac{W}{2}(156.75) + W(108) + W(36) = (730)(90)$$

$$156.75\,W + 216\,W + 72\,W = 2(730)(90)$$

$$444.75\,W = 131400$$

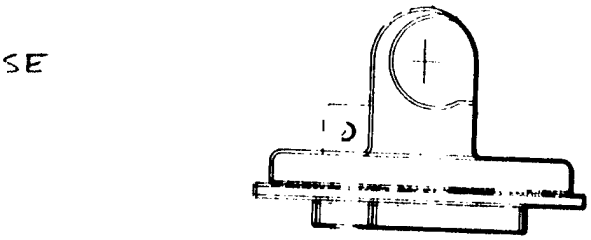
$$W = 295 \text{ lbs}$$

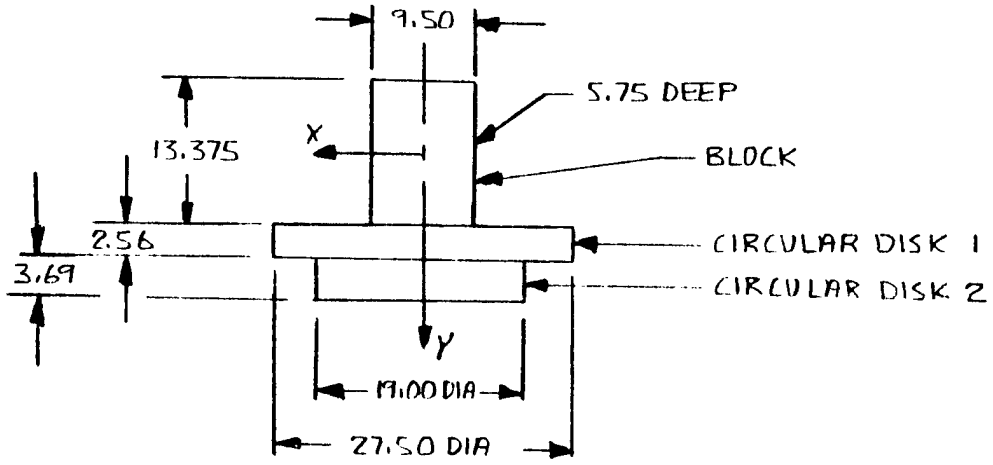
ROUND UP $W \cong 300$ lbs

AFT ARM WEIGHT = 300 lbs
FORE ARM WEIGHT = 300 lbs
NOZZLE WEIGHT = 150 lbs

LINK 1
ROTATING BASE

MATERIAL STEEL
DENSITY 0.2833 $\frac{lbs}{in^3}$





BLOCK VOLUME = (13.75)(9.50)(5.75) = 751.09 IN^3

WEIGHT = (751.09)(0.2833) = 212.78 lbs

MASS = 212.78 / 386.4 = 0.55 $\frac{lbs\text{-}sec^2}{in}$

$I_Y = \frac{1}{12}(0.55)(9.50^2 + 5.75^2) = 5.65$ lbs-sec^2-IN

CIRCULAR DISK 1

VOLUME = $\pi r^2 h = \pi (13.75)^2 (2.56) = 1520.53$ IN^3

WEIGHT = (1520.53)(0.2833) = 430.76 lbs

MASS = 430.76 / 386.4 = 1.115 $\frac{lbs\text{-}sec^2}{IN}$

$I_Y = \frac{1}{2} m r^2 = \frac{1}{2}(1.115)(13.75)^2 = 105.4$ lbs-sec^2-IN

CIRCULAR DISK 2

VOLUME = $\pi r^2 h = \pi (9.5)^2 (3.69) = 1046.72$ IN^3

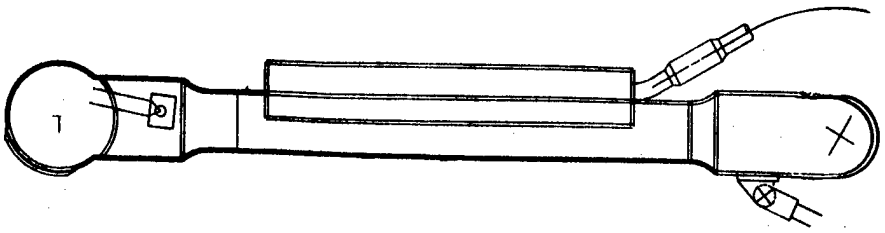
WEIGHT = (1046.72)(0.2833) = 296.39 lbs

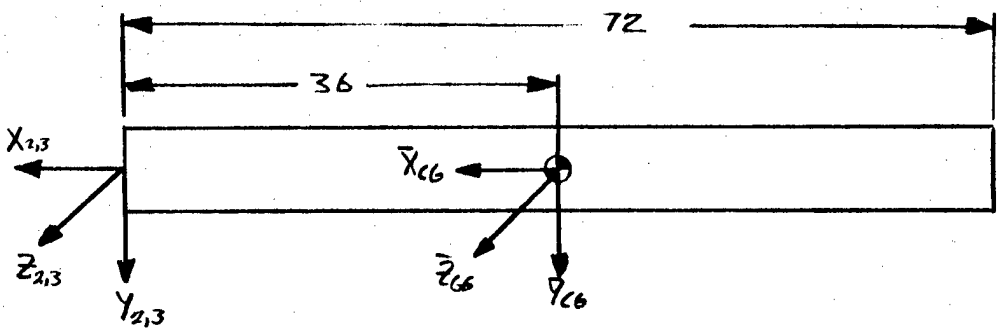
MASS = 296.39 / 386.4 = 0.767 $\frac{lbs\text{-}sec^2}{IN}$

$I_Y = \frac{1}{2} m r^2 = \frac{1}{2}(0.767)(9.50)^2 = 34.61$ lbs-sec^2-IN

TOTAL $I_Y = 5.65 + 105.4 + 34.61 = \underline{145.66}$ lbs-sec^2-IN

LINK 2
LINK 3
AFT ARM
FORE ARM





CIRCULAR CYLINDER 7.75 DIA.

CIRCULAR CYLINDER

WEIGHT = 300 lbs

MASS = 300 / 386.4 = 0.776 $\frac{lbs\text{-}sec^2}{in}$

$$\bar{I}_x = \frac{1}{2} mr^2 = \frac{1}{2}(0.776)(3.875)^2 = 5.83 \text{ lbs-sec}^2\text{-in}$$

$$\bar{I}_y = \bar{I}_z = \frac{1}{12} m(3d^2 + L^2) = \frac{1}{12}(0.776)(3(3.875)^2 + 72^2) = 338.15 \text{ lbs-sec}^2\text{-in}$$

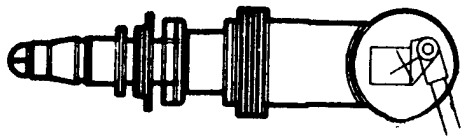
MOVE TO COORDINATE SYSTEM 2,3

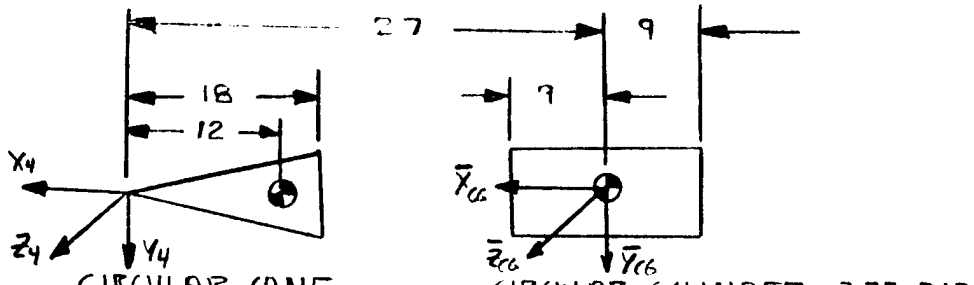
$$I_x = \bar{I}_x = 5.83 \text{ lbs-sec}^2\text{-in}$$

$$I_y = I_z = \bar{I} + md^2 = 338.15 + (0.776)(36)^2 = \underline{1344} \text{ lbs-sec-in}$$

LINK 4

NOZZLE





CIRCULAR CONE

CIRCULAR CYLINDER 7.75 DIA

CIRCULAR CYLINDER

VOLUME $= \pi r^2 h = \pi (3.875)^2 (18) = 849 \text{ IN}^3 \longrightarrow \frac{3}{4}$ TOTAL VOLUME

WEIGHT $= (0.75)(150) = 112.5$ lbs

MASS $= 112.5/386.4 = 0.291 \frac{\text{lbs-sec}^2}{\text{IN}}$

$\bar{I}_x = \frac{1}{2} m r^2 = \frac{1}{2} 0.291 (3.875)^2 = 2.185 \text{ lbs-sec}^2\text{-IN}$

$\bar{I}_y = \bar{I}_z = \frac{1}{2} m (3r^2 + L^2) = \frac{1}{2} (0.291)(3(3.875)^2 + (18)^2) = 53.696 \text{ lbs-sec}^2\text{-IN}$

MOVE TO COORDINATE SYSTEM 4

$I_x = \bar{I}_x = 2.185$

$I_y = I_z = \bar{I}_y + m d^2 = 53.696 + (0.291)(27)^2 = 336.55 \text{ lbs-sec}^2\text{-IN}$

CIRCULAR CONE (AT CS # 4)

VOLUME $= \frac{1}{3} \pi r^2 h = \frac{1}{3} \pi (3.875)^2 (18) = 283 \text{ IN}^3 \longrightarrow \frac{1}{4}$ TOTAL VOLUME

WEIGHT $= (0.25)(150) = 37.5$ lbs

MASS $= 37.5/386.4 = 0.097 \frac{\text{lbs-sec}^2}{\text{IN}}$

$I_x = \frac{3}{10} m r^2 = \frac{3}{10} (0.097)(3.385)^2 = 0.44 \text{ lbs-sec}^2\text{-IN}$

$I_y = I_z = \frac{1}{17} m (3r^2 + L^2) = \frac{1}{12} (0.097)(3(3.875)^2 + (18)^2) = 2.98 \text{ lbs-sec}^2\text{-IN}$

TOTAL

$I_x = 2.185 + 0.44 = \underline{2.62} \text{ lbs-sec}^2\text{-IN}$

$I_y = I_z = 336.55 + 2.98 = \underline{339.53} \text{ lbs-sec}^2\text{-IN}$

WEIGHT $= 150$ lbs

MASS $= 150/386.4 = \underline{0.388} \frac{\text{lbs-sec}^2}{\text{IN}}$

$\bar{X} = \frac{(37.5)(12) + (112.5)(27)}{150} = \underline{23.25} \text{ IN}$

APPENDIX C

GLOBAL "PD" CONTROLLER SIMULATION PROGRAM

```
CONTROL.C
DEF_AB.C
DEF_JH.C
DEF_JHD.C
DEF_RHS.C
DEF_W.C
DEF_W1.C
DEF_W2.C
DEF_W3.C
DEF_W4.C
DEF_WTIL.C
DIFFEQ.C
INV_4X4M.C
INV_KIN.C
KIN.C
MAIN.C
MATRIX.C
OUTPUT.C
PLOT1.C
PLOT2.C
RK4_STEP.C
TRAJ.C
```

```
/* control:  Global Controller for robot arms                              */

/*-------------------------------------------------------------------------*/
/*                                                                         */
/*  Written By:  James A. Aardema                                          */
/*                                                                         */
/*  Date:  November 28, 1988                                               */
/*                                                                         */
/*  Modifications:                                                         */
/*                                                                         */
/*  Called by:  Main                                                       */
/*                                                                         */
/*  Language: C                                                            */
/*                                                                         */
/*  Compiler Options:  None                                                */
/*                                                                         */
/*  Machine Dependencies:  None                                            */
/*                                                                         */
/*  Error:  None                                                           */
/*                                                                         */
/*  Purpose:  Implement the global position and velocity controller        */
/*                                                                         */
/*-------------------------------------------------------------------------*/

/*---Header and Include Files--------------------------------------------*/
/*                                                                       */
/*---Symbolic Constants-------------------------------------------------*/
/*                                                                       */
/*-----------------------------------------------------------------------*/

control ( m, h, t, tend, k1, k2, q, qd )
double h, t, tend, k1, k2, q[4], qd[4];
int m;
{

/*---Passed Variables---------------------------------------------------*/
/*                                                                       */
/*  m..........Number of equations to be integrated                      */
/*  h..........Integration Step Size                                     */
/*  t..........Simulation time                                           */
/*  tend.......End of Simulation Time                                    */
/*  k1.........Position Feedback gain                                    */
/*  k2.........Velocity Feedback gain                                    */
/*  q..........Joint Positions                                           */
/*  qd.........Joint Velocity                                            */
/*                                                                       */
/*-----------------------------------------------------------------------*/

  double y_d[4], yd_d[4], ydd_d[4];
  double qdd[4];
```

```
  double y[4], yd[4];
  double pos_error[4], vel_error[4];
  double Alpha[4], Beta[4][4], v[4];
  double X[8], work[50];

/*---Local Variables-------------------------------------------------------*/
/*                                                                         */
/*  y_d.........Desired global position                                    */
/*  yd_d........Desired global velcity                                     */
/*  ydd_d.......Desired global acceleration                                */
/*  qdd.........Joint Accelerations                                        */
/*  y...........Actual Position                                            */
/*  yd..........Actual Veloctiy                                           */
/*  pos_error...Position Error - ( Desired - Actual )                      */
/*  vel_error...Velocity Error - ( Desired - Actual )                      */
/*  Alpha.......Control Matrix                                             */
/*  Beta........Control Matrix                                             */
/*  v...........Control Input                                              */
/*  X...........Vector of State Equations to be Integrated;                */
/*              Initial Conditions on entry --- Final Conditions on return */
/*  work........Working Vector for the Integration routine                 */
/*                                                                         */
/*-------------------------------------------------------------------------*/

  extern double u[4];

/*---External Variables----------------------------------------------------*/
/*                                                                         */
/*  u...........Input Torques                                              */
/*                                                                         */
/*-------------------------------------------------------------------------*/

/*---Begin Control Loop */

  while ( t <= tend )
  {

/*---Get desired and actual positions, velocities and accelerations */

    trajectory ( t, y_d, yd_d, ydd_d );     /* Get desired position */
    kinematics ( q, qd, y, yd );            /* Get actual pos and vel */

/*---Calculate position and velocity error; ( desired - actual ) */

    sub_matrix (  y_d, 4, 1,  y, pos_error );
    sub_matrix ( yd_d, 4, 1, yd, vel_error );

/*---Multiply errors by proper gains */

    smult_matrix ( pos_error, 4, 1, pos_error, k1 );
    smult_matrix ( vel_error, 4, 1, vel_error, k2 );
```

```
/*---Calculate control input */

    add_matrix ( pos_error, 4, 1, vel_error, v );
    add_matrix ( v, 4, 1, ydd_d, v );

/*---Define Alpha and Beta Matrices */

    def_Alpha_Beta ( q, qd, Alpha, Beta );

/*---Caculate input Torques */

    mult_matrix ( Beta, 4, 4, v, 1, u );
    add_matrix ( Alpha, 4, 1, u, u );

/*---Call routines to output information and plotting data  */
/*   But first devide the position error by k1 and          */
/*   divide the velcity error by k2 for proper reporting    */

    smult_matrix ( pos_error, 4, 1, pos_error, 1.0/k1 );
    smult_matrix ( vel_error, 4, 1, vel_error, 1.0/k2 );

/*    output ( t, y_d, y, yd_d, yd, q, qd, pos_error, vel_error, v, u ); */
      plot1  ( t, y_d, y, yd_d, yd, q, qd, pos_error, vel_error, v, u );
/*    plot2  ( t, y_d, y, yd_d, yd, q, qd, pos_error, vel_error, v, u ); */

/*---Set up initial conditions for integrating */

    X[0] =  q[0];  X[1] =  q[1];  X[2] =  q[2];  X[3] =  q[3];
    X[4] = qd[0];  X[5] = qd[1];  X[6] = qd[2];  X[7] = qd[3];

    rk4_step ( m, X, work, &h, &t );

/*---Get joint positions and velocities from the integration routine */

    q[0] = X[0];      qd[0] = X[4];
    q[1] = X[1];      qd[1] = X[5];
    q[2] = X[2];      qd[2] = X[6];
    q[3] = X[3];      qd[3] = X[7];
  }
}
```

```
/* def_Alpha_Beta:  Defines the Alpha and Beta Matrix                    */

/*--------------------------------------------------------------------------*/
/*                                                                          */
/*  Written By:  James A. Aardema                                           */
/*                                                                          */
/*  Date:  November 26, 1988                                                */
/*                                                                          */
/*  Modifications:                                                          */
/*                                                                          */
/*  Called by:                                                              */
/*                                                                          */
/*  Language: C                                                             */
/*                                                                          */
/*  Compiler Options:  None                                                 */
/*                                                                          */
/*  Machine Dependencies:                                                   */
/*                                                                          */
/*  Error:                                                                  */
/*                                                                          */
/*  Purpose:                                                                */
/*                                                                          */
/*--------------------------------------------------------------------------*/

/*---Header and Include Files---------------------------------------------*/
/*                                                                          */
/*---Symbolic Constants---------------------------------------------------*/
/*                                                                          */
/*--------------------------------------------------------------------------*/

def_Alpha_Beta ( q, qd, Alpha, Beta )
double q[4], qd[4], Alpha[4], Beta[4][4];
{

/*---Passed Variables-----------------------------------------------------*/
/*                                                                          */
/*  q...........Joint Positions                                             */
/*  qd..........Joint Velocities                                            */
/*  Alpha.......Alpha Matrix                                                */
/*  Beta........Beta Matrix                                                 */
/*                                                                          */
/*--------------------------------------------------------------------------*/

  double W[4][4], Wtilde[4][4], Jh[4][4], Jh_inv[4][4], Jhd[4][4];
  double Wtilde_tp[4][4], tempm[4][4], tempv1[4], tempv2[4];
  int    error;

/*---Local Variables------------------------------------------------------*/
/*                                                                          */
/*  W...........Weight matrix W                                             */
```

```
/*  Wtilde......W tilde matrix                                              */
/*  Jh..........Jacobian Matrix                                             */
/*  Jh_inv......Inverse of the Jacobian Matrix                              */
/*  Jhd.........Derivative of the Jacobian Matrix                           */
/*  Wtilde_tp...Transpose of the W tilde Matrix                             */
/*  tempm.......Temporary Matrix                                            */
/*  tempv1......Temporary Vector 1                                          */
/*  tempv2......Temporary Vector 2                                          */
/*  error.......Error Flag for the Matrix Inversion Routine                 */
/*                                                                          */
/*--------------------------------------------------------------------------*/

/*---External Variables-----------------------------------------------------*/
/*                                                                          */
/*--------------------------------------------------------------------------*/

  def_w ( W, q );
  def_Wtilde ( Wtilde, q, qd );
  def_Jh ( q, Jh );

  inv_4x4matrix ( Jh, Jh_inv, error );

  def_Jhd ( q, qd, Jhd );

  mult_matrix ( W, 4, 4, Jh_inv, 4, Beta );

  mult_matrix ( Beta, 4, 4, Jhd, 4, tempm );
  mult_matrix ( tempm, 4, 4, qd, 1, tempv2 );

  transpose_matrix ( Wtilde, 4, 4, Wtilde_tp );
  smult_matrix ( Wtilde, 4, 4, tempm, 0.5 );
  sub_matrix ( Wtilde_tp, 4, 4, tempm, tempm );
  mult_matrix ( tempm, 4, 4, qd, 1, tempv1 );
  sub_matrix ( tempv1, 4, 1, tempv2, Alpha );

}
```

```
/* def_Jh: Defines the Jacobian Martix                                    */

/*------------------------------------------------------------------------*/
/*                                                                        */
/*  Written By:  James A. Aardema                                         */
/*                                                                        */
/*  Date:  November 10, 1988                                              */
/*                                                                        */
/*  Modifications:                                                        */
/*                                                                        */
/*  Called by:                                                            */
/*                                                                        */
/*  Language: C                                                           */
/*                                                                        */
/*  Compiler Options:  None                                               */
/*                                                                        */
/*  Purpose:                                                              */
/*                                                                        */
/*------------------------------------------------------------------------*/

def_Jh ( q, Jh )
double q[4], Jh[4][4];
{

/*---Passed Variables---------------------------------------------------*/
/*                                                                      */
/*  q..........Joint Positions                                          */
/*  Jh.........Jacobian Matrix                                          */
/*                                                                      */
/*----------------------------------------------------------------------*/

  double c1, s1, c2, s2, c23, s23, c234, s234;

/*---Local Variables----------------------------------------------------*/
/*                                                                      */
/*  c1,   s1....Cosine and Sine of angle 1 - Waist Angle                */
/*  c2,   s2....Cosine and Sine of angle 2 - Shoulder Angle             */
/*  c23,  s23...Cosine and Sine of ( angle 2 + angle 3 )                */
/*  c234, s234..Cosine and Sine of ( angle 2 + angle 3 + angle 4 )      */
/*                                                                      */
/*----------------------------------------------------------------------*/

  extern double sin(), cos();

/*---External Variables-------------------------------------------------*/
/*                                                                      */
/*  sin().......Sine of an angle                                        */
/*  cos().......Cosine of an angle                                      */
/*                                                                      */
/*----------------------------------------------------------------------*/
```

```
/*  Remember that arrays start at zero */
/*  q[0] = Waist Angle                 */
/*  q[1] = Shoulder Angle              */
/*  q[2] = Elbow Angle                 */
/*  q[3] = Wrist Angle                 */

/*  Calculate some local variables */

  c1 = cos ( q[0] );                  s1 = sin ( q[0] );
  c2 = cos ( q[1] );                  s2 = sin ( q[1] );

  c23  = cos ( q[1] + q[2] );         s23 = sin ( q[1] + q[2] );

  c234 = cos ( q[1] + q[2] + q[3] );  s234 = sin ( q[1] + q[2] + q[3] );

/* The first row */

  Jh[0][0] = -36.0*s1*c234 - 72.0*s1*c23 - 72.0*s1*c2;
  Jh[0][1] = -36.0*c1*s234 - 72.0*c1*s23 - 72.0*c1*s2;
  Jh[0][2] = -36.0*c1*s234 - 72.0*c1*s23;
  Jh[0][3] = -36.0*c1*s234;

/* The second row */

  Jh[1][0] =  36.0*c1*c234 + 72.0*c1*c23 + 72.0*c1*c2;
  Jh[1][1] = -36.0*s1*s234 - 72.0*s1*s23 - 72.0*s1*s2;
  Jh[1][2] = -36.0*s1*s234 - 72.0*s1*s23;
  Jh[1][3] = -36.0*s1*s234;

/* The third row */

  Jh[2][0] =  0.0;
  Jh[2][1] = -36.0*c234 - 72.0*c23 - 72.0*c2;
  Jh[2][2] = -36.0*c234 - 72.0*c23;
  Jh[2][3] = -36.0*c234;

/* The fourth row */

  Jh[3][0] = 0.0;
  Jh[3][1] = 1.0;
  Jh[3][2] = 1.0;
  Jh[3][3] = 1.0;

}
```

```
/* def_Jhd:  Defines the time derivative of the Jacobian Matrix            */

/*-------------------------------------------------------------------------*/
/*                                                                         */
/*  Written By:  James A. Aardema                                          */
/*                                                                         */
/*  Date:  November 10, 1988                                               */
/*                                                                         */
/*  Modifications:                                                         */
/*                                                                         */
/*  Called by:                                                             */
/*                                                                         */
/*  Language: C                                                            */
/*                                                                         */
/*  Compiler Options:  None                                                */
/*                                                                         */
/*  Purpose:                                                               */
/*                                                                         */
/*-------------------------------------------------------------------------*/

def_Jhd ( q, qd, Jhd )
double q[4], qd[4], Jhd[4][4];
{

/*---Passed Variables-----------------------------------------------------*/
/*                                                                         */
/*  q...........Joint Positions                                            */
/*  qd..........Joint Velocities                                           */
/*  Jhd.........Time derivative of the Jacobian Matrix                     */
/*                                                                         */
/*-------------------------------------------------------------------------*/

  double c1, s1, c2, s2, c23, s23, c234, s234;

/*---Local Variables------------------------------------------------------*/
/*                                                                         */
/*  c1,   s1....Cosine and Sine of angle 1 - Waist Angle                   */
/*  c2,   s2....Cosine and Sine of angle 2 - Shoulder Angle                */
/*  c23,  s23...Cosine and Sine of ( angle 2 + angle 3 )                   */
/*  c234, s234..Cosine and Sine of ( angle 2 + angle 3 + angle 4 )         */
/*                                                                         */
/*-------------------------------------------------------------------------*/

  extern double sin(), cos();

/*---External Variables---------------------------------------------------*/
/*                                                                         */
/*  sin().......Sine of an angle                                           */
/*  cos().......Cosine of an angle                                         */
/*                                                                         */
```

```
/*-------------------------------------------------------------------------*/

/*  Remember that arrays start at zero */
/*  q[0] = Waist Angle                  */
/*  q[1] = Shoulder Angle               */
/*  q[2] = Elbow Angle                  */
/*  q[3] = Wrist Angle                  */

/*  Calculate some local variables */

  c1 = cos ( q[0] );                    s1 = sin ( q[0] );
  c2 = cos ( q[1] );                    s2 = sin ( q[1] );

  c23  = cos ( q[1] + q[2] );           s23 = sin ( q[1] + q[2] );

  c234 = cos ( q[1] + q[2] + q[3] );  s234 = sin ( q[1] + q[2] + q[3] );

/* The first row */

  Jhd[0][0] = - 36.0*c1*c234*qd[0] + 36.0*s1*s234*(qd[1]+qd[2]+qd[3])
              - 72.0*c1*c23*qd[0]  + 72.0*s1*s23*(qd[1]+qd[2])
              - 72.0*c1*c2*qd[0]   + 72.0*s1*s2*qd[1];

  Jhd[0][1] =   36.0*s1*s234*qd[0] - 36.0*c1*c234*(qd[1]+qd[2]+qd[3])
              + 72.0*s1*s23*qd[0]  - 72.0*c1*c23*(qd[1]+qd[2])
              + 72.0*s1*s2*qd[0]   - 72.0*c1*c2*qd[1];

  Jhd[0][2] =   36.0*s1*s234*qd[0] - 36.0*c1*c234*(qd[1]+qd[2]+qd[3])
              + 72.0*s1*s23*qd[0]  - 72.0*c1*c23*(qd[1]+qd[2]);

  Jhd[0][3] =   36.0*s1*s234*qd[0] - 36.0*c1*c234*(qd[1]+qd[2]+qd[3]);

/* The second row */

  Jhd[1][0] = - 36.0*s1*c234*qd[0] - 36.0*c1*s234*(qd[1]+qd[2]+qd[3])
              - 72.0*s1*c23*qd[0]  - 72.0*c1*s23*(qd[1]+qd[2])
              - 72.0*s1*c2*qd[0]   - 72.0*c1*s2*qd[1];

  Jhd[1][1] = - 36.0*c1*s234*qd[0] - 36.0*s1*c234*(qd[1]+qd[2]+qd[3])
              - 72.0*c1*s23*qd[0]  - 72.0*s1*c23*(qd[1]+qd[2])
              - 72.0*c1*s2*qd[0]   - 72.0*s1*c2*qd[1];

  Jhd[1][2] = - 36.0*c1*s234*qd[0] - 36.0*s1*c234*(qd[1]+qd[2]+qd[3])
              - 72.0*c1*s23*qd[0]  - 72.0*s1*c23*(qd[1]+qd[2]);

  Jhd[1][3] = - 36.0*c1*s234*qd[0] - 36.0*s1*c234*(qd[1]+qd[2]+qd[3]);

/* The third row */

  Jhd[2][0] =   0.0;
```

```
    Jhd[2][1] =   36.0*s234*(qd[1]+qd[2]+qd[3])
                + 72.0*s23*(qd[1]+qd[2])
                + 72.0*s2*qd[1];

    Jhd[2][2] =   36.0*s234*(qd[1]+qd[2]+qd[3])
                + 72.0*s23*(qd[1]+qd[2]);

    Jhd[2][3] =   36.0*s234*(qd[1]+qd[2]+qd[3]);

  /* The fourth row */

    Jhd[3][0] =   0.0;
    Jhd[3][1] =   0.0;
    Jhd[3][2] =   0.0;
    Jhd[3][3] =   0.0;

  }
```

```
/* def_rhs_Lagrange:  Defines the Right Hand Side of the Lagrange equation */

/*-------------------------------------------------------------------------*/
/*                                                                         */
/*  Written By:  James A. Aardema                                          */
/*                                                                         */
/*  Date:  November 26, 1988                                               */
/*                                                                         */
/*  Modifications:                                                         */
/*                                                                         */
/*  Called by:                                                             */
/*                                                                         */
/*  Language: C                                                            */
/*                                                                         */
/*  Compiler Options:  None                                                */
/*                                                                         */
/*  Machine Dependencies:                                                  */
/*                                                                         */
/*  Error:                                                                 */
/*                                                                         */
/*  Purpose:                                                               */
/*                                                                         */
/*-------------------------------------------------------------------------*/

/*---Header and Include Files--------------------------------------------*/
/*                                                                       */
/*---Symbolic Constants-------------------------------------------------*/
/*                                                                       */
/*-----------------------------------------------------------------------*/

def_rhs_Lagrange ( u, q, qd, qdd )
double u[4], q[4], qd[4], qdd[4];
{

/*---Passed Variables---------------------------------------------------*/
/*                                                                       */
/*  u...........Input Torques                                            */
/*  q...........Joint Positions                                          */
/*  q...........Joint Velocities                                         */
/*  q...........Joint Accelerations                                      */
/*                                                                       */
/*-----------------------------------------------------------------------*/

  double W[4][4], Wtilde[4][4], W_inv[4][4], Wtilde_tp[4][4];
  double tempm1[4][4], tempm2[4][4], tempv1[4], tempv2[4];
  int    error;

/*---Local Variables----------------------------------------------------*/
/*                                                                       */
/*  W...........Weight Matrix                                            */
```

```
/*  Wtilde......W tilde Matrix                                          */
/*  W_inv.......Inverse of the Weight Matrix                            */
/*  Wtilde_tp...Transpose of the W tilde Matrix                         */
/*  tempm1......Tempoary Matrix 1                                       */
/*  tempm2......Tempoary Matrix 2                                       */
/*  tempv1......Tempoary Vector 1                                       */
/*  tempv2......Tempoary Vector 2                                       */
/*  error.......Error Code for the Matrix Inversion Routine             */
/*                                                                      */
/*----------------------------------------------------------------------*/

/*---External Variables-------------------------------------------------*/
/*                                                                      */
/*----------------------------------------------------------------------*/

  def_w ( W, q );                          /*  Define the W matrix      */
  def_Wtilde ( Wtilde, q, qd );            /*  Define the Wtilde Matrix */

  inv_4x4matrix ( W, W_inv, error );       /*  Invert the W matrix      */

  mult_matrix ( W_inv, 4, 4, u, 1, tempv1 );

  transpose_matrix ( Wtilde, 4, 4, Wtilde_tp );    /* Transpose W tilde */

  smult_matrix ( Wtilde, 4, 4, tempm1, 0.5 );
  sub_matrix ( Wtilde_tp, 4, 4, tempm1, tempm1 );
  mult_matrix ( W_inv, 4, 4, tempm1, 4, tempm2 );
  mult_matrix ( tempm2, 4, 4, qd, 1, tempv2 );

  sub_matrix ( tempv1, 4, 1, tempv2, qdd );

}
```

```
/* def_w:  Calculate the total Weight matrix for the arm                    */

/*-------------------------------------------------------------------------*/
/*                                                                         */
/*  Written By:  James A. Aardema                                          */
/*                                                                         */
/*  Date:  November 23, 1988                                               */
/*                                                                         */
/*  Modifications:                                                         */
/*                                                                         */
/*  Called by:                                                             */
/*                                                                         */
/*  Language: C                                                            */
/*                                                                         */
/*  Compiler Options:  None                                                */
/*                                                                         */
/*  Purpose:  This subroutine calculate the Inertial matrix of link 1      */
/*  The total Inertial matrix for the robot is the sum of each link's      */
/*  Inertial matrix.                                                       */
/*                      W = W1 + W2 + W3 + W4                              */
/*                                                                         */
/*-------------------------------------------------------------------------*/

/*---Header and Include Files--------------------------------------------*/
/*                                                                       */
/*---Symbolic Constants-------------------------------------------------*/
/*                                                                       */
/*-----------------------------------------------------------------------*/

def_w ( w, q )
double w[4][4], q[4];
{

/*---Passed Variables---------------------------------------------------*/
/*                                                                       */
/*  w...........Weight Matrix for robot arm                              */
/*  q...........Joint Positions                                          */
/*                                                                       */
/*-----------------------------------------------------------------------*/

  double wi[4][4];

/*---Local Variables----------------------------------------------------*/
/*                                                                       */
/*   wi.........Weight Matrix for Link "i"                               */
/*                                                                       */
/*-----------------------------------------------------------------------*/

/*---External Variables-------------------------------------------------*/
/*                                                                       */
```

```
/*-------------------------------------------------------------------------*/

  def_w1 (  w, q );                    /* Define W1                  */
  def_w2 ( wi, q );                    /* Define W2                  */
  add_matrix ( w, 4, 4, wi, w );       /* Add W1 + W2 = W            */
  def_w3 ( wi, q );                    /* Define W3                  */
  add_matrix ( w, 4, 4, wi, w );       /* (W1 + W2) + W3 = W         */
  def_w4 ( wi, q );                    /* Define W4                  */
  add_matrix ( w, 4, 4, wi, w );       /* (W1 + W2 + W3) + W4 = W */

}
```

```
/* def_w1:  Calculate the Weight matrix of link 1                          */

/*-------------------------------------------------------------------------*/
/*                                                                         */
/*  Written By:  James A. Aardema                                          */
/*                                                                         */
/*  Date:  November 17, 1988                                               */
/*                                                                         */
/*  Modifications:                                                         */
/*                                                                         */
/*  Called by:                                                             */
/*                                                                         */
/*  Language: C                                                            */
/*                                                                         */
/*  Compiler Options:  None                                                */
/*                                                                         */
/*  Purpose:  This subroutine calculate the Inertial matrix of link 1      */
/*  The total Inertial matrix for the robot is the sum of each link's      */
/*  Inertial matrix.                                                       */
/*                      W = W1 + W2 + W3 + W4                              */
/*                                                                         */
/*-------------------------------------------------------------------------*/

def_w1 ( w, q )
double w[4][4], q[4];
{

/*---Passed Variables-----------------------------------------------------*/
/*                                                                         */
/*  w...........Weight Matrix for link 1                                   */
/*  q...........Joint Positions                                            */
/*                                                                         */
/*-------------------------------------------------------------------------*/

/*---Local Variables------------------------------------------------------*/
/*                                                                         */
/*-------------------------------------------------------------------------*/

  extern double Cx1, Cy1, Cz1, m1, Ixx1, Iyy1, Izz1, Ixy1, Ixz1, Iyz1;

/*---External Variables---------------------------------------------------*/
/*                                                                         */
/*  Cx1.........X distance from CS to CG                                   */
/*  Cy1.........Y distance from CS to CG                                   */
/*  Cz1.........Z distance from CS to CG                                   */
/*  m1..........Mass                                                       */
/*  Ixx1........Moment of Inertia                                          */
/*  Iyy1........                                                           */
/*  Izz1........                                                           */
/*  Ixy1........Product of Inertia                                         */
```

```
/*  Ixz1........                                                          */
/*  Iyz1........                                                          */
/*                                                                        */
/*------------------------------------------------------------------------*/

  zero_matrix ( w, 4, 4 );
  w[0][0] = Iyy1;

}
```

```
/* def_w2:  Calculate the Weight matrix W2 of link 2                    */

/*--------------------------------------------------------------------*/
/*                                                                    */
/*  Written By:  James A. Aardema                                     */
/*                                                                    */
/*  Date:  November 17, 1988                                          */
/*                                                                    */
/*  Modifications:                                                    */
/*                                                                    */
/*  Called by:                                                        */
/*                                                                    */
/*  Language: C                                                       */
/*                                                                    */
/*  Compiler Options:  None                                           */
/*                                                                    */
/*  Purpose:  This subroutine calculate the Inertial matrix of link 2 */
/*  The total Inertial matrix for the robot is the sum of each link's */
/*  Inertial matrix.                                                  */
/*                      W = W1 + W2 + W3 + W4                         */
/*                                                                    */
/*--------------------------------------------------------------------*/

/*---Header and Include Files-----------------------------------------*/
/*                                                                    */
/*---Symbolic Constants-----------------------------------------------*/
/*                                                                    */
/*--------------------------------------------------------------------*/

def_w2 ( w, q )
double w[4][4], q[4];
{

/*---Passed Variables-------------------------------------------------*/
/*                                                                    */
/*  w...........Weight Matrix                                         */
/*  q...........Joint Positions                                       */
/*                                                                    */
/*--------------------------------------------------------------------*/

  double U[6][6], J[6][4], Jt[4][6], temp[4][6];
  double c2, s2;

/*---Local Variables--------------------------------------------------*/
/*                                                                    */
/*  U...........Inertial Matrix                                       */
/*  J...........Sub Jacobian Matix                                    */
/*  Jt..........Transpose of the Sub Jacobian Matrix                  */
/*  temp........Temporary Matrix                                      */
/*  c2,   s2....Cosine and Sine of angle 2 - Shoulder Angle           */
```

```
/*                                                                      */
/*----------------------------------------------------------------------*/

  extern double Cx2, Cy2, Cz2, m2, Ixx2, Iyy2, Izz2, Ixy2, Ixz2, Iyz2;
  extern double sin(), cos();

/*---External Variables-------------------------------------------------*/
/*                                                                      */
/*  Cx2.........X distance from CS to CG                                */
/*  Cy2.........Y distance from CS to CG                                */
/*  Cz2.........Z distance from CS to CG                                */
/*  m2..........Mass                                                    */
/*  Ixx2........Moment of Inertia                                       */
/*  Iyy2........                                                        */
/*  Izz2........                                                        */
/*  Ixy2........Product of Inertia                                      */
/*  Ixz2........                                                        */
/*  Iyz2........                                                        */
/*  sin().......Sine of an angle                                        */
/*  cos().......Cosine of an angle                                      */
/*                                                                      */
/*----------------------------------------------------------------------*/

/*  q[0] = Waist Angle                  */
/*  q[1] = Shoulder Angle               */
/*  q[2] = Elbow Angle                  */
/*  q[3] = Wrist Angle                  */

/*  Calculate some local variables */

  c2 = cos ( q[1] );
  s2 = sin ( q[1] );

/*  Define the U matrix */

  U[0][0] =      m2;   U[0][1] =     0.0;   U[0][2] =     0.0;
  U[1][0] =     0.0;   U[1][1] =      m2;   U[1][2] =     0.0;
  U[2][0] =     0.0;   U[2][1] =     0.0;   U[2][2] =      m2;

  U[3][0] =     0.0;   U[3][1] = -Cz2*m2;   U[3][2] =  Cy2*m2;
  U[4][0] =  Cz2*m2;   U[4][1] =     0.0;   U[4][2] = -Cx2*m2;
  U[5][0] = -Cy2*m2;   U[5][1] =  Cx2*m2;   U[5][2] =     0.0;

  U[0][3] =     0.0;   U[0][4] =  Cz2*m2;   U[0][5] = -Cy2*m2;
  U[1][3] = -Cz2*m2;   U[1][4] =     0.0;   U[1][5] =  Cx2*m2;
  U[2][3] =  Cy2*m2;   U[2][4] = -Cx2*m2;   U[2][5] =     0.0;

  U[3][3] =    Ixx2;   U[3][4] =    Ixy2;   U[3][5] =    Ixz2;
  U[4][3] =    Ixy2;   U[4][4] =    Iyy2;   U[4][5] =    Iyz2;
  U[5][3] =    Ixz2;   U[5][4] =    Iyz2;   U[5][5] =    Izz2;
```

```
/*  Define the Sub Jacobian Matrix for Link 2 */

  zero_matrix ( J, 6, 4 );

  J[2][0] =  72.0*c2;
  J[3][0] = -s2;
  J[4][0] = -c2;

  J[1][1] =  72.0;
  J[5][1] =   1.0;

/* Calculate the W matrix */

  transpose_matrix ( J, 6, 4, Jt );
  mult_matrix ( Jt, 4, 6, U, 6, temp );
  mult_matrix ( temp, 4, 6, J, 4, w );

}
```

```
/* def_w3:  Calculate the Weight matrix W3 of link 3                        */

/*--------------------------------------------------------------------------*/
/*                                                                          */
/*  Written By:  James A. Aardema                                           */
/*                                                                          */
/*  Date:  November 17, 1988                                                */
/*                                                                          */
/*  Modifications:                                                          */
/*                                                                          */
/*  Called by:                                                              */
/*                                                                          */
/*  Language: C                                                             */
/*                                                                          */
/*  Compiler Options:  None                                                 */
/*                                                                          */
/*  Purpose:  This subroutine calculate the Inertial matrix of link 3       */
/*  The total Inertial matrix for the robot is the sum of each link's       */
/*  Inertial matrix.                                                        */
/*                      W = W1 + W2 + W3 + W4                               */
/*                                                                          */
/*--------------------------------------------------------------------------*/

/*---Header and Include Files----------------------------------------------*/
/*                                                                          */
/*---Symbolic Constants----------------------------------------------------*/
/*                                                                          */
/*--------------------------------------------------------------------------*/

def_w3 ( w, q )
double w[4][4], q[4];
{

/*---Passed Variables------------------------------------------------------*/
/*                                                                          */
/*  w...........Weight Matrix                                               */
/*  q...........Joint Positions                                             */
/*                                                                          */
/*--------------------------------------------------------------------------*/

  double U[6][6], J[6][4], Jt[4][6], temp[4][6];
  double c2, s2, c3, s3, c23, s23;

/*---Local Variables-------------------------------------------------------*/
/*                                                                          */
/*  U...........Inertial Matrix                                             */
/*  J...........Sub Jacobian Matix                                          */
/*  Jt..........Transpose of the sub Jacobian Matrix                        */
/*  temp........Temporary Matrix                                            */
/*  c2,   s2....Cosine and Sine of angle 2 - Shoulder Angle                 */
```

```
/*  c3,   s3....Cosine and Sine of angle 3 - Elbow Angle                    */
/*  c23,  s23...Cosine and Sine of ( angle 2 + angle 3 )                    */
/*                                                                          */
/*--------------------------------------------------------------------------*/

  extern double Cx3, Cy3, Cz3, m3, Ixx3, Iyy3, Izz3, Ixy3, Ixz3, Iyz3;
  extern double sin(), cos();

/*---External Variables---------------------------------------------------*/
/*                                                                          */
/*  Cx3.........X distance from CS to CG                                    */
/*  Cy3.........Y distance from CS to CG                                    */
/*  Cz3.........Z distance from CS to CG                                    */
/*  m3..........Mass                                                        */
/*  Ixx3........Moment of Inertia                                           */
/*  Iyy3........                                                            */
/*  Izz3........                                                            */
/*  Ixy3........Product of Inertia                                          */
/*  Ixz3........                                                            */
/*  Iyz3........                                                            */
/*  sin().......Sine of an angle                                            */
/*  cos().......Cosine of an angle                                          */
/*                                                                          */
/*--------------------------------------------------------------------------*/

/*  q[0] = Waist Angle                  */
/*  q[1] = Shoulder Angle               */
/*  q[2] = Elbow Angle                  */
/*  q[3] = Wrist Angle                  */

/*  Calculate some local variables */

  c2 = cos ( q[1] );
  s2 = sin ( q[1] );
  c3 = cos ( q[2] );
  s3 = sin ( q[2] );

  c23 = cos ( q[1] + q[2] );
  s23 = sin ( q[1] + q[2] );

/*  Define the U matrix */

  U[0][0] =      m3;   U[0][1] =     0.0;   U[0][2] =     0.0;
  U[1][0] =     0.0;   U[1][1] =      m3;   U[1][2] =     0.0;
  U[2][0] =     0.0;   U[2][1] =     0.0;   U[2][2] =      m3;

  U[3][0] =     0.0;   U[3][1] = -Cz3*m3;   U[3][2] =  Cy3*m3;
  U[4][0] =  Cz3*m3;   U[4][1] =     0.0;   U[4][2] = -Cx3*m3;
  U[5][0] = -Cy3*m3;   U[5][1] =  Cx3*m3;   U[5][2] =     0.0;

  U[0][3] =     0.0;   U[0][4] =  Cz3*m3;   U[0][5] = -Cy3*m3;
```

```
  U[1][3] = -Cz3*m3;   U[1][4] =     0.0;   U[1][5] =  Cx3*m3;
  U[2][3] =  Cy3*m3;   U[2][4] = -Cx3*m3;   U[2][5] =     0.0;

  U[3][3] =    Ixx3;   U[3][4] =    Ixy3;   U[3][5] =    Ixz3;
  U[4][3] =    Ixy3;   U[4][4] =    Iyy3;   U[4][5] =    Iyz3;
  U[5][3] =    Ixz3;   U[5][4] =    Iyz3;   U[5][5] =    Izz3;

/*  Define the Sub Jacobian Matrix for Link 3 */

  zero_matrix ( J, 6, 4 );

  J[2][0] =  72.0*c23 + 72.0*c2;
  J[3][0] = -s23;
  J[4][0] = -c23;

  J[0][1] =  72.0*s3;
  J[1][1] =  72.0 + 72.0*c3;
  J[5][1] =  1.0;

  J[1][2] =  72.0;
  J[5][2] =  1.0;

/* Calculate the W matrix */

  transpose_matrix ( J, 6, 4, Jt );
  mult_matrix ( Jt, 4, 6, U, 6, temp );
  mult_matrix ( temp, 4, 6, J, 4, w );

}
```

```
/* def_w4:  Calculate the Weight matrix of link 4                          */

/*-------------------------------------------------------------------------*/
/*                                                                         */
/*  Written By:  James A. Aardema                                          */
/*                                                                         */
/*  Date:  November 17, 1988                                               */
/*                                                                         */
/*  Modifications:                                                         */
/*                                                                         */
/*  Called by:                                                             */
/*                                                                         */
/*  Language: C                                                            */
/*                                                                         */
/*  Compiler Options:  None                                                */
/*                                                                         */
/*  Purpose:  This subroutine calculate the Inertial matrix of link 4      */
/*  The total Inertial matrix for the robot is the sum of each link's      */
/*  Inertial matrix.                                                       */
/*                      W = W1 + W2 + W3 + W4                              */
/*                                                                         */
/*-------------------------------------------------------------------------*/

/*---Header and Include Files--------------------------------------------*/
/*                                                                       */
/*---Symbolic Constants-------------------------------------------------*/
/*                                                                       */
/*-----------------------------------------------------------------------*/

def_w4 ( w, q )
double w[4][4], q[4];
{

/*---Passed Variables---------------------------------------------------*/
/*                                                                       */
/*  w...........Weight Matrix                                            */
/*  q...........Joint Positions                                          */
/*                                                                       */
/*-----------------------------------------------------------------------*/

  double U[6][6], J[6][4], Jt[4][6], temp[4][6];
  double c2, s2, c3, s3, c23, s23, c4, s4, c34, s34, c234, s234;

/*---Local Variables----------------------------------------------------*/
/*                                                                       */
/*  U...........Inertial Matrix                                          */
/*  J...........Sub Jacobian Matix                                       */
/*  Jt..........Transpose of the Sub Jacobian Matrix                     */
/*  temp........Temporary Matrix                                         */
/*  c2,   s2....Cosine and Sine of angle 2 - Shoulder Angle              */
```

```
/*  c3,   s3....Cosine and Sine of angle 3 - Elbow Angle                  */
/*  c4,   s4....Cosine and Sine of angle 4 - Wrist Angle                  */
/*  c23,  s23...Cosine and Sine of ( angle 2 + angle 3 )                  */
/*  c34,  s34...Cosine and Sine of ( angle 3 + angle 4 )                  */
/*  c234, s234..Cosine and Sine of ( angle 2 + angle 3 + angle 4 )        */
/*                                                                        */
/*------------------------------------------------------------------------*/

  extern double Cx4, Cy4, Cz4, m4, Ixx4, Iyy4, Izz4, Ixy4, Ixz4, Iyz4;
  extern double sin(), cos();

/*---External Variables--------------------------------------------------*/
/*                                                                        */
/*  Cx4.........X distance from CS to CG                                  */
/*  Cy4.........Y distance from CS to CG                                  */
/*  Cz4.........Z distance from CS to CG                                  */
/*  m4..........Mass                                                      */
/*  Ixx4........Moment of Inertia                                         */
/*  Iyy4........                                                          */
/*  Izz4........                                                          */
/*  Ixy4........Product of Inertia                                        */
/*  Ixz4........                                                          */
/*  Iyz4........                                                          */
/*  sin().......Sine of an angle                                          */
/*  cos().......Cosine of an angle                                        */
/*                                                                        */
/*------------------------------------------------------------------------*/

/*  q[0] = Waist Angle                  */
/*  q[1] = Shoulder Angle               */
/*  q[2] = Elbow Angle                  */
/*  q[3] = Wrist Angle                  */

/*  Calculate some local variables */

  c2 = cos ( q[1] );
  s2 = sin ( q[1] );
  c3 = cos ( q[2] );
  s3 = sin ( q[2] );
  c4 = cos ( q[3] );
  s4 = sin ( q[4] );

  c23 = cos ( q[1] + q[2] );
  s23 = sin ( q[1] + q[2] );

  c34 = cos ( q[2] + q[3] );
  s34 = sin ( q[2] + q[3] );

  c234 = cos ( q[1] + q[2] + q[3] );
  s234 = sin ( q[1] + q[2] + q[3] );
```

```
/*  Define the U matrix */

  U[0][0] =      m4;   U[0][1] =     0.0;   U[0][2] =     0.0;
  U[1][0] =     0.0;   U[1][1] =      m4;   U[1][2] =     0.0;
  U[2][0] =     0.0;   U[2][1] =     0.0;   U[2][2] =      m4;

  U[3][0] =     0.0;   U[3][1] = -Cz4*m4;   U[3][2] =  Cy4*m4;
  U[4][0] =  Cz4*m4;   U[4][1] =     0.0;   U[4][2] = -Cx4*m4;
  U[5][0] = -Cy4*m4;   U[5][1] =  Cx4*m4;   U[5][2] =     0.0;

  U[0][3] =     0.0;   U[0][4] =  Cz4*m4;   U[0][5] = -Cy4*m4;
  U[1][3] = -Cz4*m4;   U[1][4] =     0.0;   U[1][5] =  Cx4*m4;
  U[2][3] =  Cy4*m4;   U[2][4] = -Cx4*m4;   U[2][5] =     0.0;

  U[3][3] =    Ixx4;   U[3][4] =    Ixy4;   U[3][5] =    Ixz4;
  U[4][3] =    Ixy4;   U[4][4] =    Iyy4;   U[4][5] =    Iyz4;
  U[5][3] =    Ixz4;   U[5][4] =    Iyz4;   U[5][5] =    Izz4;

/*  Define the Sub Jacobian Matrix for Link 3 */

  zero_matrix ( J, 6, 4 );

  J[2][0] =  36.0*c234 + 72.0*c23 + 72.0*c2;
  J[3][0] = -s234;
  J[4][0] = -c234;

  J[0][1] =  72.0*s4 + 72.0*s34;
  J[1][1] =  36.0 + 72.0*c4 + 72.0*c34;
  J[5][1] =  1.0;

  J[0][2] =  72.0*s4;
  J[1][2] =  36.0 + 72.0*c4;
  J[5][2] =  1.0;

  J[1][3] =  36.0;
  J[5][3] =  1.0;

/* Calculate the W matrix  W = J' U J */

  transpose_matrix ( J, 6, 4, Jt );
  mult_matrix ( Jt, 4, 6, U, 6, temp );
  mult_matrix ( temp, 4, 6, J, 4, w );

}
```

```
/* def_Wtilde:  Defines the W tilde Weight Matrix                          */

/*-------------------------------------------------------------------------*/
/*                                                                         */
/*  Written By:  James A. Aardema                                          */
/*                                                                         */
/*  Date:  November 28, 1988                                               */
/*                                                                         */
/*  Modifications:                                                         */
/*                                                                         */
/*  Called by:                                                             */
/*                                                                         */
/*  Language: C                                                            */
/*                                                                         */
/*  Compiler Options:  None                                                */
/*                                                                         */
/*  Purpose: Define the W tilde matrix which is used in the                */
/*           Lagrange Equation                                             */
/*                                                                         */
/*-------------------------------------------------------------------------*/

def_Wtilde ( Wtilde, q, qd )
double Wtilde[4][4], q[4], qd[4];
{

/*---Passed Variables-----------------------------------------------------*/
/*                                                                         */
/*  Wtilde......W tilde matrix                                             */
/*  q...........Joint Positions                                            */
/*  qd..........Joint Velocities                                           */
/*                                                                         */
/*-------------------------------------------------------------------------*/

  double c1, s1, c2, s2, c3, s3, c4, s4, c23, s23, c34, s34, c234, s234;
  double w2q211;
  double w3q211;
  double w3q311, w3q322, w3q323, w3q332;
  double w4q211;
  double w4q311, w4q322, w4q323, w4q324, w4q332, w4q342;
  double w4q411, w4q422, w4q423, w4q424, w4q432, w4q433, w4q434,
         w4q442, w4q443;

/*---Local Variables------------------------------------------------------*/
/*                                                                         */
/*  c1,   s1....Cosine and Sine of angle 1 - Waist Angle                   */
/*  c2,   s2....Cosine and Sine of angle 2 - Shoulder Angle                */
/*  c3,   s3....Cosine and Sine of angle 3 - Elbow Angle                   */
/*  c4,   s4....Cosine and Sine of angle 4 - Wrist Angle                   */
/*  c23,  s23...Cosine and Sine of ( angle 2 + angle 3 )                   */
/*  c34,  s34...Cosine and Sine of ( angle 3 + angle 4 )                   */
```

```
/*  c234, s234..Cosine and Sine of ( angle 2 + angle 3 + angle 4 )          */
/*                                                                          */
/*  w2q211......Partial derivative of W2 wrt to q2 - element (1,1)          */
/*                                                                          */
/*  w3q211......Partial derivative of W3 wrt to q2 - element (1,1)          */
/*                                                                          */
/*  w3q311......Partial derivative of W3 wrt to q3 - element (1,1)          */
/*  w3q322......Partial derivative of W3 wrt to q3 - element (2,2)          */
/*  w3q323......Partial derivative of W3 wrt to q3 - element (2,3)          */
/*  w3q332......Partial derivative of W3 wrt to q3 - element (3,2)          */
/*                                                                          */
/*  w4q211......Partial derivative of W4 wrt to q2 - element (1,1)          */
/*                                                                          */
/*  w4q311......Partial derivative of W4 wrt to q3 - element (1,1)          */
/*  w4q322......Partial derivative of W4 wrt to q3 - element (2,2)          */
/*  w4q323......Partial derivative of W4 wrt to q3 - element (2,3)          */
/*  w4q324......Partial derivative of W4 wrt to q3 - element (2,4)          */
/*  w4q332......Partial derivative of W4 wrt to q3 - element (3,2)          */
/*  w4q342......Partial derivative of W4 wrt to q3 - element (4,2)          */
/*                                                                          */
/*  w4q411......Partial derivative of W4 wrt to q4 - element (1,1)          */
/*  w4q422......Partial derivative of W4 wrt to q4 - element (2,2)          */
/*  w4q423......Partial derivative of W4 wrt to q4 - element (2,3)          */
/*  w4q424......Partial derivative of W4 wrt to q4 - element (2,4)          */
/*  w4q432......Partial derivative of W4 wrt to q4 - element (3,2)          */
/*  w4q433......Partial derivative of W4 wrt to q4 - element (3,3)          */
/*  w4q434......Partial derivative of W4 wrt to q4 - element (3,4)          */
/*  w4q442......Partial derivative of W4 wrt to q4 - element (4,2)          */
/*  w4q443......Partial derivative of W4 wrt to q4 - element (4,3)          */
/*                                                                          */
/*--------------------------------------------------------------------------*/

  extern double Cx1, Cy1, Cz1, m1, Ixx1, Iyy1, Izz1, Ixy1, Ixz1, Iyz1;
  extern double Cx2, Cy2, Cz2, m2, Ixx2, Iyy2, Izz2, Ixy2, Ixz2, Iyz2;
  extern double Cx3, Cy3, Cz3, m3, Ixx3, Iyy3, Izz3, Ixy3, Ixz3, Iyz3;
  extern double Cx4, Cy4, Cz4, m4, Ixx4, Iyy4, Izz4, Ixy4, Ixz4, Iyz4;
  extern double sin(), cos();

/*---External Variables---------------------------------------------------*/
/*                                                                          */
/*  Cx.........X distance from CS to CG                                     */
/*  Cy.........Y distance from CS to CG                                     */
/*  Cz.........Z distance from CS to CG                                     */
/*  m..........Mass                                                         */
/*  Ixx........Moment of Inertia                                            */
/*  Iyy........                                                             */
/*  Izz........                                                             */
/*  Ixy........Product of Inertia                                           */
/*  Ixz........                                                             */
/*  Iyz........                                                             */
/*  sin().....Sine of an angle                                              */
```

```
/*  cos().......Cosine of an angle                                          */
/*                                                                          */
/*--------------------------------------------------------------------------*/

/*  Remember that arrays start at zero */
/*  q[0] = Waist Angle                  */
/*  q[1] = Shoulder Angle               */
/*  q[2] = Elbow Angle                  */
/*  q[3] = Wrist Angle                  */

/*  Calculate some local variables */

  c1 = cos ( q[0] );                 s1 = sin ( q[0] );
  c2 = cos ( q[1] );                 s2 = sin ( q[1] );
  c3 = cos ( q[2] );                 s3 = sin ( q[2] );
  c4 = cos ( q[3] );                 s4 = sin ( q[3] );

  c23  = cos ( q[1] + q[2] );        s23 = sin ( q[1] + q[2] );
  c34  = cos ( q[2] + q[3] );        s34 = sin ( q[2] + q[3] );

  c234 = cos ( q[1] + q[2] + q[3] );  s234 = sin ( q[1] + q[2] + q[3] );

/*---Calculate the partial derivatives of W2 wrt q2 */

  w2q211 = - 2.0 * 72.0 * 72.0 * c2 * s2  * m2
           - 4.0 * 72.0 * c2   * s2 * Cx2 * m2
           + 2.0 * Ixx2 * s2   * c2
           - 2.0 * Iyy2 * c2   * s2;

/*---Calculate the partial derivates of W3 wrt q2 */

  w3q211 = - 2.0 * 72.0 * 72.0 * c23 * s23 * m3
           - 2.0 * 72.0 * 72.0 * s23 * c2  * m3
           - 2.0 * 72.0 * 72.0 * c23 * s2  * m3
           - 2.0 * 72.0 * 72.0 * c2  * s2  * m3
           - 4.0 * 72.0 * c23  * s23 * Cx3 * m3
           - 2.0 * 72.0 * s23  * c2  * Cx3 * m3
           - 2.0 * 72.0 * c23  * s2  * Cx3 * m3
           + 2.0 * Ixx3 * s23  * c23
           - 2.0 * Iyy3 * c23  * s23;

/*---Calculate the partial derivates of W3 wrt q3 */

  w3q311 = - 2.0 * 72.0 * 72.0 * c23 * s23 * m3
           - 2.0 * 72.0 * 72.0 * s23 * c2  * m3
           - 4.0 * 72.0 * c23  * s23 * Cx3 * m3
           - 2.0 * 72.0 * s23  * c2  * Cx3 * m3
           + 2.0 * Ixx3 * s23  * c23
           - 2.0 * Iyy3 * c23  * s23;

  w3q322 = - 2.0 * 72.0 * 72.0 * s3  * m3
```

```
          - 2.0 * 72.0 * s3   * Cx3 * m3;

  w3q323 = - 72.0 * 72.0 * s3  * m3
           - 72.0 * s3   * Cx3 * m3;

  w3q332 = w3q323;

/*---Caculate the partial derivatives of W4 wrt q2 */

  w4q211 = - 2.0 * 36.0 * 36.0 * c234 * s234 * m4
           - 2.0 * 36.0 * 72.0 * s23  * c234 * m4
           - 2.0 * 36.0 * 72.0 * c23  * s234 * m4
           - 2.0 * 36.0 * 72.0 * s2   * c234 * m4
           - 2.0 * 36.0 * 72.0 * c2   * s234 * m4
           - 2.0 * 72.0 * 72.0 * c23  * s23  * m4
           - 2.0 * 72.0 * 72.0 * s2   * c23  * m4
           - 2.0 * 72.0 * 72.0 * c2   * s23  * m4
           - 2.0 * 72.0 * 72.0 * c2   * s2   * m4
           - 4.0 * 36.0 * c234 * s234 * Cx4  * m4
           - 2.0 * 72.0 * s23  * c234 * Cx4  * m4
           - 2.0 * 72.0 * c23  * s234 * Cx4  * m4
           - 2.0 * 72.0 * s2   * c234 * Cx4  * m4
           - 2.0 * 72.0 * c2   * s234 * Cx4  * m4
           + 2.0 * Ixx4 * s234 * c234
           - 2.0 * Iyy4 * c234 * s234;

/*---Calculate the partial derivatives of W4 wrt q3 */

  w4q311 = - 2.0 * 36.0 * 36.0 * c234 * s234 * m4
           - 2.0 * 36.0 * 72.0 * s23  * c234 * m4
           - 2.0 * 36.0 * 72.0 * c23  * s234 * m4
           - 2.0 * 36.0 * 72.0 * c2   * s234 * m4
           - 2.0 * 72.0 * 72.0 * c23  * s23  * m4
           - 2.0 * 72.0 * 72.0 * c2   * s23  * m4
           - 4.0 * 36.0 * c234 * s234 * Cx4  * m4
           - 2.0 * 72.0 * s23  * c234 * Cx4  * m4
           - 2.0 * 72.0 * c23  * s234 * Cx4  * m4
           - 2.0 * 72.0 * c2   * s234 * Cx4  * m4
           + 2.0 * Ixx4 * s234 * c234
           - 2.0 * Iyy4 * c234 * s234;

  w4q322 = + 2.0 * 72.0 * 72.0 * s4  * c34 * m4
           - 2.0 * 36.0 * 72.0 * s34 * m4
           - 2.0 * 72.0 * 72.0 * s34 * c4  * m4
           - 2.0 * 72.0 * s34  * Cx4 * m4;

  w4q323 = + 72.0 * 72.0 * s4  * c34 * m4
           - 72.0 * 72.0 * c4  * s34 * m4
           - 36.0 * 72.0 * s34 * m4
           - 72.0 * s34  * Cx4 * m4;
```

```
    w4q324 = - 36.0 * 72.0 * s34 * m4
             - 72.0 * s34  * Cx4 * m4;

    w4q332 = w4q323;

    w4q342 = w4q324;

  /*---Calculate the partial derivatives of W4 wrt q4 */

    w4q411 = - 2.0 * 36.0 * 36.0 * c234 * s234 * m4
             - 2.0 * 36.0 * 72.0 * c23  * s234 * m4
             - 2.0 * 36.0 * 72.0 * c2   * s234 * m4
             - 4.0 * 36.0 * c234 * s234 * Cx4  * m4
             - 2.0 * 72.0 * c23  * s234 * Cx4  * m4
             - 2.0 * 72.0 * c2   * s234 * Cx4  * m4
             + 2.0 * Ixx4 * s234 * c234
             - 2.0 * Iyy4 * c234 * s234;

    w4q422 =   2.0 * 72.0 * 72.0 * c4  * s34 * m4
             + 2.0 * 72.0 * 72.0 * s4  * c34 * m4
             - 2.0 * 36.0 * 72.0 * s4  * m4
             - 2.0 * 36.0 * 72.0 * s34 * m4
             - 2.0 * 72.0 * 72.0 * s34 * c4 * m4
             - 2.0 * 72.0 * 72.0 * c34 * s4 * m4
             - 2.0 * 72.0 * s4   * Cx4 * m4
             - 2.0 * 72.0 * s34  * Cx4 * m4;

    w4q423 =   72.0 * 72.0 * c4   * s34 * m4
             + 72.0 * 72.0 * s4   * c34 * m4
             - 72.0 * 72.0 * s4   * c34 * m4
             - 72.0 * 72.0 * c4   * s34 * m4
             -  2.0 * 36.0 * 72.0 * s4  * m4
             -  2.0 * 72.0 * s4   * Cx4 * m4
             - 36.0 * 72.0 * s34  * m4
             - 72.0 * s34  * Cx4  * m4;

    w4q424 = - 36.0 * 72.0 * s4  * m4
             - 36.0 * 72.0 * s34 * m4
             - 72.0 * s4   * Cx4 * m4
             - 72.0 * s34  * Cx4 * m4;

    w4q432 = w4q423;

    w4q433 = - 2.0 * 36.0 * 72.0 * s4  * m4
             - 2.0 * 72.0 * s4   * Cx4 * m4;

    w4q434 = - 36.0 * 72.0 * s4  * m4
             - 72.0 * s4   * Cx4 * m4;

    w4q442 = w4q424;
```

```
  w4q443 = w4q434;

/*---Define the W tilde Matrix */

/* Row 1 */

  Wtilde[0][0] = 0.0;
  Wtilde[0][1] = 0.0;
  Wtilde[0][2] = 0.0;
  Wtilde[0][3] = 0.0;

/* Row 2 */

  Wtilde[1][0] = qd[0] * ( w2q211 + w3q211 + w4q211 );
  Wtilde[1][1] = 0.0;
  Wtilde[1][2] = 0.0;
  Wtilde[1][3] = 0.0;

/* Row 3 */

  Wtilde[2][0] = qd[0] * ( w3q311 + w4q311 );
  Wtilde[2][1] = qd[1] * ( w3q322 + w4q322 )
               + qd[2] * ( w3q332 + w4q332 )
               + qd[3] * ( w4q342 );
  Wtilde[2][2] = qd[1] * ( w3q323 + w4q323 );
  Wtilde[2][3] = qd[1] * ( w4q324 );

/* Row 4 */

  Wtilde[3][0] = qd[0] * ( w4q411 );
  Wtilde[3][1] = qd[1] * w4q422 + qd[2] * w4q432 + qd[3] * w4q442;
  Wtilde[3][2] = qd[1] * w4q423 + qd[2] * w4q433 + qd[3] * w4q443;
  Wtilde[3][3] = qd[1] * w4q424 + qd[2] * w4q434;

}
```

```
/* diffeq:  Sets the differential equations for the robot arm              */

/*-------------------------------------------------------------------------*/
/*                                                                         */
/*  Written By:  James A. Aardema                                          */
/*                                                                         */
/*  Date:  November 28, 1988                                               */
/*                                                                         */
/*  Modifications:                                                         */
/*                                                                         */
/*  Called by:  rk4_step:  Integration Routine                             */
/*                                                                         */
/*  Language: C                                                            */
/*                                                                         */
/*  Compiler Options:  None                                                */
/*                                                                         */
/*  Machine Dependencies: None                                             */
/*                                                                         */
/*  Error:  None                                                           */
/*                                                                         */
/*  Purpose:  Sets up the differential equations for the integration       */
/*  routine.  This routine call def_rhs_Lagrange to get the right hand     */
/*  side of the Lagrange differential equations.                           */
/*                                                                         */
/*-------------------------------------------------------------------------*/

/*---Header and Include Files--------------------------------------------*/
/*                                                                       */
/*---Symbolic Constants--------------------------------------------------*/
/*                                                                       */
/*-----------------------------------------------------------------------*/

diffeq ( f, t, y, m )
double f[], t, y[];
int m;
{

/*---Passed Variables----------------------------------------------------*/
/*                                                                       */
/*  f...........First derivative of the state equations                  */
/*  t...........Current Time                                             */
/*  y...........Current Value of the state equations                     */
/*  m...........Number of differential equations to be integrated        */
/*                                                                       */
/*-----------------------------------------------------------------------*/

  double q[4], qd[4], qdd[4];

/*---Local Variables-----------------------------------------------------*/
/*                                                                       */
```

```
/*  q...........Joint Positions                                          */
/*  qd..........Joint Velocities                                         */
/*  qdd.........Joint Accelerations                                      */
/*                                                                       */
/*---------------------------------------------------------------------*/

  extern double u[4];

/*---External Variables------------------------------------------------*/
/*                                                                       */
/*  u...........Input Torques                                            */
/*                                                                       */
/*---------------------------------------------------------------------*/

/* Get the current Joint Positions */

  q[0] = y[0];        /* y[0] position of joint 1 */
  q[1] = y[1];        /* y[1] position of joint 2 */
  q[2] = y[2];        /* y[2] position of joint 3 */
  q[3] = y[3];        /* y[3] position of joint 4 */

/* Get the current Joint Velocities */

  qd[0] = y[4];       /* y[4] velocity of joint 1 */
  qd[1] = y[5];       /* y[5] velocity of joint 2 */
  qd[2] = y[6];       /* y[6] velocity of joint 3 */
  qd[3] = y[7];       /* y[7] velocity of joint 4 */

/* Get the Joint Accelerations from the r.h.s of the Lagrange Equation */

  def_rhs_Lagrange ( u, q, qd, qdd );

/*---Differential Equations to be integrated---------------------------*/

/*---Integrate Velocity to get position */

  f[0] = qd[0];         /* Velocity of Joint 1 */
  f[1] = qd[1];         /* Velocity of Joint 2 */
  f[2] = qd[2];         /* Velocity of Joint 3 */
  f[3] = qd[3];         /* Velocity of Joint 4 */

/*---Integrate Acceleration to get velocity */

  f[4] = qdd[0];        /* Accel of joint 1 */
  f[5] = qdd[1];        /* Accel of Joint 2 */
  f[6] = qdd[2];        /* Accel of Joint 3 */
  f[7] = qdd[3];        /* Accel of Joint 4 */
}
```

```
/* inv_4x4matrix:  Inverts a 4x4 matrix                                    */

/*-------------------------------------------------------------------------*/
/*                                                                         */
/*  Written By:  James A. Aardema                                          */
/*                                                                         */
/*  Date:  November 24, 1988                                               */
/*                                                                         */
/*  Modifications:                                                         */
/*                                                                         */
/*  Called by:                                                             */
/*                                                                         */
/*  Language: C                                                            */
/*                                                                         */
/*  Error:  Returns and error code of 1 if the Determinant is zero         */
/*          and set the inverse matrix to all zeros.                       */
/*                                                                         */
/*  Compiler Options:  None                                                */
/*                                                                         */
/*  Purpose:  This subroutine calculate the inverse of a 4x4 matrix        */
/*  using a brute force technique.                                         */
/*                                                                         */
/*                      -1     adj ( A  )                                  */
/*                     A   =  -----------                                  */
/*                                 det A                                   */
/*-------------------------------------------------------------------------*/

/*---Header and Include Files--------------------------------------------*/
/*                                                                       */
/*---Symbolic Constants-------------------------------------------------*/
/*                                                                       */
/*-----------------------------------------------------------------------*/

inv_4x4matrix ( a, ai, error )
double a[4][4], ai[4][4];
int *error;
{

/*---Passed Variables---------------------------------------------------*/
/*                                                                       */
/*  a...........Matrix A                                                 */
/*  ai..........Matrix A Inverse                                         */
/*  error.......Error Code - Set to 1 if determinant is zero; 0 otherwise */
/*                                                                       */
/*-----------------------------------------------------------------------*/

  double det;
  register int i, j;

/*---Local Variables----------------------------------------------------*/
```

```
/*                                                                          */
/*  det.........Determinant of a Matrix                                     */
/*  i...........Index                                                       */
/*  j...........Index                                                       */
/*                                                                          */
/*--------------------------------------------------------------------------*/

  double extern fabs ();

/*---External Variables---------------------------------------------------*/
/*                                                                          */
/*  fabs........Floating Point Absolute value                               */
/*                                                                          */
/*--------------------------------------------------------------------------*/

/*  Calculate the Determinant */

  *error = 0;

  det =   a[0][0] * (   a[1][1] * ( a[2][2]*a[3][3] - a[3][2]*a[2][3] )
                      - a[2][1] * ( a[1][2]*a[3][3] - a[3][2]*a[1][3] )
                      + a[3][1] * ( a[1][2]*a[2][3] - a[2][2]*a[1][3] ) )

        - a[1][0] * (   a[0][1] * ( a[2][2]*a[3][3] - a[3][2]*a[2][3] )
                      - a[2][1] * ( a[0][2]*a[3][3] - a[3][2]*a[0][3] )
                      + a[3][1] * ( a[0][2]*a[2][3] - a[2][2]*a[0][3] ) )

        + a[2][0] * (   a[0][1] * ( a[1][2]*a[3][3] - a[3][2]*a[1][3] )
                      - a[1][1] * ( a[0][2]*a[3][3] - a[3][2]*a[0][3] )
                      + a[3][1] * ( a[0][2]*a[1][3] - a[1][2]*a[0][3] ) )

        - a[3][0] * (   a[0][1] * ( a[1][2]*a[2][3] - a[2][2]*a[1][3] )
                      - a[1][1] * ( a[0][2]*a[2][3] - a[2][2]*a[0][3] )
                      + a[2][1] * ( a[0][2]*a[1][3] - a[1][2]*a[0][3] ) );

  if ( fabs ( det ) < 1.0E-008 )
  {
    printf ("\nERROR in Matrix Inversion Algorithm\n");
    printf ("      No unique solution exists\n");
    printf ("      Determinant nearly zero ( less than 1.0E-008 )\n");
    printf ("      Inverse Matrix will be set to all zeros\n");
    *error = 1;
    for ( i = 0; i < 4; i++ )
      for ( j = 0; j < 4; j++ )
      {
        ai[i][j] = 0.0;                    /* Zero Inverse Matrix */
      }
    goto end;
  }

/*  The first row */
```

```
  ai[0][0] =   (  a[1][1] * ( a[2][2]*a[3][3] - a[2][3]*a[3][2] )
                 -a[1][2] * ( a[2][1]*a[3][3] - a[2][3]*a[3][1] )
                 +a[1][3] * ( a[2][1]*a[3][2] - a[2][2]*a[3][1] ) ) / det;

  ai[0][1] = - (  a[0][1] * ( a[2][2]*a[3][3] - a[2][3]*a[3][2] )
                 -a[0][2] * ( a[2][1]*a[3][3] - a[2][3]*a[3][1] )
                 +a[0][3] * ( a[2][1]*a[3][2] - a[2][2]*a[3][1] ) ) / det;

  ai[0][2] =   (  a[0][1] * ( a[1][2]*a[3][3] - a[1][3]*a[3][2] )
                 -a[0][2] * ( a[1][1]*a[3][3] - a[1][3]*a[3][1] )
                 +a[0][3] * ( a[1][1]*a[3][2] - a[1][2]*a[3][1] ) ) / det;

  ai[0][3] = - (  a[0][1] * ( a[1][2]*a[2][3] - a[1][3]*a[2][2] )
                 -a[0][2] * ( a[1][1]*a[2][3] - a[1][3]*a[2][1] )
                 +a[0][3] * ( a[1][1]*a[2][2] - a[1][2]*a[2][1] ) ) / det;

/*  The second row */

  ai[1][0] = - (  a[1][0] * ( a[2][2]*a[3][3] - a[2][3]*a[3][2] )
                 -a[1][2] * ( a[2][0]*a[3][3] - a[2][3]*a[3][0] )
                 +a[1][3] * ( a[2][0]*a[3][2] - a[2][2]*a[3][0] ) ) / det;

  ai[1][1] =   (  a[0][0] * ( a[2][2]*a[3][3] - a[2][3]*a[3][2] )
                 -a[0][2] * ( a[2][0]*a[3][3] - a[2][3]*a[3][0] )
                 +a[0][3] * ( a[2][0]*a[3][2] - a[2][2]*a[3][0] ) ) / det;

  ai[1][2] = - (  a[0][0] * ( a[1][2]*a[3][3] - a[1][3]*a[3][2] )
                 -a[0][2] * ( a[1][0]*a[3][3] - a[1][3]*a[3][0] )
                 +a[0][3] * ( a[1][0]*a[3][2] - a[1][2]*a[3][0] ) ) / det;

  ai[1][3] =   (  a[0][0] * ( a[1][2]*a[2][3] - a[1][3]*a[2][2] )
                 -a[0][2] * ( a[1][0]*a[2][3] - a[1][3]*a[2][0] )
                 +a[0][3] * ( a[1][0]*a[2][2] - a[1][2]*a[2][0] ) ) / det;

/*  The third row */

  ai[2][0] =   (  a[1][0] * ( a[2][1]*a[3][3] - a[2][3]*a[3][1] )
                 -a[1][1] * ( a[2][0]*a[3][3] - a[2][3]*a[3][0] )
                 +a[1][3] * ( a[2][0]*a[3][1] - a[2][1]*a[3][0] ) ) / det;

  ai[2][1] = - (  a[0][0] * ( a[2][1]*a[3][3] - a[2][3]*a[3][1] )
                 -a[0][1] * ( a[2][0]*a[3][3] - a[2][3]*a[3][0] )
                 +a[0][3] * ( a[2][0]*a[3][1] - a[2][1]*a[3][0] ) ) / det;

  ai[2][2] =   (  a[0][0] * ( a[1][1]*a[3][3] - a[1][3]*a[3][1] )
                 -a[0][1] * ( a[1][0]*a[3][3] - a[1][3]*a[3][0] )
                 +a[0][3] * ( a[1][0]*a[3][1] - a[1][1]*a[3][0] ) ) / det;

  ai[2][3] = - (  a[0][0] * ( a[1][1]*a[2][3] - a[1][3]*a[2][1] )
                 -a[0][1] * ( a[1][0]*a[2][3] - a[1][3]*a[2][0] )
```

```
                +a[0][3] * ( a[1][0]*a[2][1] - a[1][1]*a[2][0] ) ) / det;

/*  The fourth row */

  ai[3][0] = - (  a[1][0] * ( a[2][1]*a[3][2] - a[2][2]*a[3][1] )
                 -a[1][1] * ( a[2][0]*a[3][2] - a[2][2]*a[3][0] )
                 +a[1][2] * ( a[2][0]*a[3][1] - a[2][1]*a[3][0] ) ) / det;

  ai[3][1] =    (  a[0][0] * ( a[2][1]*a[3][2] - a[2][2]*a[3][1] )
                 -a[0][1] * ( a[2][0]*a[3][2] - a[2][2]*a[3][0] )
                 +a[0][2] * ( a[2][0]*a[3][1] - a[2][1]*a[3][0] ) ) / det;

  ai[3][2] = - (  a[0][0] * ( a[1][1]*a[3][2] - a[1][2]*a[3][1] )
                 -a[0][1] * ( a[1][0]*a[3][2] - a[1][2]*a[3][0] )
                 +a[0][2] * ( a[1][0]*a[3][1] - a[1][1]*a[3][0] ) ) / det;

  ai[3][3] =    (  a[0][0] * ( a[1][1]*a[2][2] - a[1][2]*a[2][1] )
                 -a[0][1] * ( a[1][0]*a[2][2] - a[1][2]*a[2][0] )
                 +a[0][2] * ( a[1][0]*a[2][1] - a[1][1]*a[2][0] ) ) / det;

  end:
;
}
```

```
/* inv_kin:  Inverse Kinematics                                          */

/*-----------------------------------------------------------------------*/
/*                                                                       */
/*  Written By:  James A. Aardema                                        */
/*                                                                       */
/*  Date:  November 26, 1988                                             */
/*                                                                       */
/*  Modifications:                                                       */
/*                                                                       */
/*  Called by:                                                           */
/*                                                                       */
/*  Language: C                                                          */
/*                                                                       */
/*  Compiler Options:  None                                              */
/*                                                                       */
/*  Machine Dependencies:                                                */
/*                                                                       */
/*  Error:                                                               */
/*                                                                       */
/*  Purpose:                                                             */
/*                                                                       */
/*-----------------------------------------------------------------------*/

/*---Header and Include Files--------------------------------------------*/
/*                                                                       */
/*---Symbolic Constants-------------------------------------------------*/
/*                                                                       */
/*-----------------------------------------------------------------------*/

inv_kin ( y, yd, q, qd )
double y[4], yd[4], q[4], qd[4];
{

/*---Passed Variables---------------------------------------------------*/
/*                                                                       */
/*  y..........Global (Cartesian) Position of nozzle                     */
/*  yd.........Global (Cartesian) Velocity of nozzle                     */
/*  q..........Joint Positions                                           */
/*  qd.........Joint Velocities                                          */
/*                                                                       */
/*-----------------------------------------------------------------------*/

  double Jh[4][4], Jh_inv[4][4];
  double c1, s1, c2, s2, ca, sa;
  double p3xg, p3yg, p3zg;
  double p3x1, p3y1, p3z1;
  double r, salpha, calpha, sbeta, cbeta, sgamma, cgamma;

/*---Local Variables----------------------------------------------------*/
```

```
/*                                                                      */
/*  Jh.........Jacobian Matrix                                          */
/*  Jh_inv......Inverse of the Jacobian Matrix                          */
/*  c1.........Cosine of Angle 1; Waist Joint                           */
/*  s1.........Sine of Angle 1; Waist Joint                             */
/*  c2.........Cosine of Angle 2; Shoulder Joint                        */
/*  s2.........Sine of Angle 2; Shoulder Joint                          */
/*  ca.........Cosine of the Approach Angle                             */
/*  sa.........Sine of the Approach Angle                               */
/*  p3xg........Global X distance to CS 3                               */
/*  p3yg........Global Y distance to CS 3                               */
/*  p3zg........Global Z distance to CS 3                               */
/*  p3x1........X distance from CS 1 to CS 3                            */
/*  p3y1........Y distance from CS 1 to CS 3                            */
/*  p3z1........Z distance from CS 1 to CS 3                            */
/*  r...........Distance from CS 1 to CS 3 in the X1,Y1 plane           */
/*  salpha......Sine of Alpha                                           */
/*  calpha......Cosine of Alpha                                         */
/*  sbeta.......Sine of Beta                                            */
/*  cbeta.......Cosine of Beta                                          */
/*  sgamma......Sine of Gamma                                           */
/*  cgamma......Cosine of Gamma                                         */
/*                                                                      */
/*----------------------------------------------------------------------*/

  extern double atan2(), cos(), sin(), sqrt();

/*---External Variables-------------------------------------------------*/
/*                                                                      */
/*  atan2.......Arctan                                                  */
/*  sin.........Sine of an Angle                                        */
/*  cos.........Cosine of an Angle                                      */
/*  sqrt........Square Root                                             */
/*                                                                      */
/*----------------------------------------------------------------------*/

/*  Remember that arrays start at zero */
/*  q[0] = Waist Angle                 */
/*  q[1] = Shoulder Angle              */
/*  q[2] = Elbow Angle                 */
/*  q[3] = Wrist Angle                 */

/*---Define some local variables */

  ca = cos ( y[3] );
  sa = sin ( y[3] );

/*---Calculate the angle of the rotating base;  Joint 1;  q[0]  */

  q[0] = atan2 ( y[1], y[0] );
  c1 = cos ( q[0] );
```

```
  s1 = sin ( q[0] );

/*---Calculate the global position of Coordinate System 3 */

  p3xg = y[0] - 36.0*c1*ca;
  p3yg = y[1] - 36.0*s1*ca;
  p3zg = y[2] + 36.0*sa;

/*---Calculate the position of CS 3 with respect to CS 1 */

  p3x1 =  p3xg*c1 + p3yg*s1;
  p3y1 = -p3zg + 11.25;
  p3z1 = -p3xg*s1 + p3yg*c1;

/*---Calculate the distance from CS 1 to CS3 */

  r = sqrt ( p3x1*p3x1 + p3y1*p3y1 );

/*---Calculate the angle of the shoulder;  Joint 2;  q[1]  */

  salpha = p3y1 / r;
  calpha = p3x1 / r;

  cbeta = r * r / ( 144.0 * r );
  sbeta = sqrt ( 1.0 - cbeta*cbeta );

  s2 = salpha*cbeta - calpha*sbeta;
  c2 = calpha*cbeta + salpha*sbeta;

  q[1] = atan2 ( s2, c2 );

/*---Calculate the angle of the elbow;  Joint 3;  q[2] */

  cgamma = ( -(r*r) + 10368.0 ) / 10368.0;
  sgamma = sqrt ( 1.0 - cgamma*cgamma );

  q[2] = 3.141592654 - atan2 ( sgamma, cgamma );

/*---Calculate the angle of the wrist;  Joint4;  q[3] */

  q[3] = y[3] - q[2] - q[1];

/*---Calculate the joint velocities */

  def_Jh ( q, Jh );
  inv_4x4matrix ( Jh, Jh_inv );

  mult_matrix ( Jh_inv, 4, 4, yd, 1, qd );
}
```

```
/* kinematics:  Calculate global position and velocity given joint angle   */
/*              and joint velocities                                       */

/*-------------------------------------------------------------------------*/
/*                                                                         */
/*  Written By:  James A. Aardema                                          */
/*                                                                         */
/*  Date:  November 10, 1988                                               */
/*                                                                         */
/*  Modifications:                                                         */
/*                                                                         */
/*  Called by:                                                             */
/*                                                                         */
/*  Language: C                                                            */
/*                                                                         */
/*  Compiler Options:  None                                                */
/*                                                                         */
/*  Purpose:                                                               */
/*  q[0] = Waist Angle                                                     */
/*  q[1] = Shoulder Angle                                                  */
/*  q[2] = Elbow Angle                                                     */
/*  q[3] = Wrist Angle                                                     */
/*                                                                         */
/*-------------------------------------------------------------------------*/

kinematics ( q, qd, y, yd )
double q[4], qd[4], y[4], yd[4];
{

/*---Passed Variables-----------------------------------------------------*/
/*                                                                         */
/*  q...........Joint Positions                                            */
/*  qd..........Joint Velocities                                           */
/*  y...........Global Position of nozzle                                  */
/*  yd..........Global Velocity of nozzle                                  */
/*                                                                         */
/*-------------------------------------------------------------------------*/

  double Jh[4][4];
  double c1, s1, c2, s2, c23, s23, c234, s234;

/*---Local Variables------------------------------------------------------*/
/*                                                                         */
/*  Jh..........Jacobian Matix                                             */
/*  c1,   s1....Cosine and Sine of angle 1 - Waist Angle                   */
/*  c2,   s2....Cosine and Sine of angle 2 - Shoulder Angle                */
/*  c23,  s23...Cosine and Sine of ( angle 2 + angle 3 )                   */
/*  c234, s234..Cosine and Sine of ( angle 2 + angle 3 + angle 4 )         */
/*                                                                         */
/*-------------------------------------------------------------------------*/
```

```
  extern double sin(), cos();

/*---External Variables-------------------------------------------------------*/
/*                                                                            */
/*  sin().......Sine of an angle                                              */
/*  cos().......Cosine of an angle                                            */
/*                                                                            */
/*----------------------------------------------------------------------------*/

  /*  Remember that arrays start at zero */
  /*  q[0] = Waist Angle                 */
  /*  q[1] = Shoulder Angle              */
  /*  q[2] = Elbow Angle                 */
  /*  q[3] = Wrist Angle                 */

  /*  Calculate some local variables */

    c1 = cos ( q[0] );                    s1 = sin ( q[0] );
    c2 = cos ( q[1] );                    s2 = sin ( q[1] );

    c23  = cos ( q[1] + q[2] );           s23 = sin ( q[1] + q[2] );
    c234 = cos ( q[1] + q[2] + q[3] );    s234 = sin ( q[1] + q[2] + q[3] );

  /*  Calculate the global position of the nozzle */

    y[0] =  36.0*c1*c234 + 72.0*c1*c23 + 72.0*c1*c2;
    y[1] =  36.0*s1*c234 + 72.0*s1*c23 + 72.0*s1*c2;
    y[2] = -36.0*s234    - 72.0*s23    - 72.0*s2 + 11.25;
    y[3] =  q[1] + q[2] + q[3];

  /*  Define the Jacobian Matrix */

    def_Jh ( q, Jh );

  /*  Calculate the nozzle velocity using yd = Jh*qd  */

    mult_matrix ( Jh, 4, 4, qd, 1, yd );

  }
```

```
/* main:  Main Entry upon Start of Execution                              */
/*------------------------------------------------------------------------*/
/*                                                                        */
/*  Written By:  James A. Aardema                                         */
/*                                                                        */
/*  Date:  November 29, 1988                                              */
/*                                                                        */
/*  Modifications:                                                        */
/*                                                                        */
/*  Called by:                                                            */
/*                                                                        */
/*  Language: C                                                           */
/*                                                                        */
/*  Compiler Options:  None                                               */
/*                                                                        */
/*  Machine Dependencies:  None                                           */
/*                                                                        */
/*  Error:  None                                                          */
/*                                                                        */
/*  Purpose:  Start Execution of program, Initialize variables, and start */
/*            control process                                             */
/*                                                                        */
/*------------------------------------------------------------------------*/

/*---Header and Include Files--------------------------------------------*/
/*                                                                        */
#include <math.h>
#include <stdio.h>
/*                                                                        */
/*---Symbolic Constants--------------------------------------------------*/
/*                                                                        */
/*------------------------------------------------------------------------*/

/*---Initialize Rotating Base (Link 1) Parameters */

double Cx1  =    0.000;     /* X distance from CS to CG */
double Cy1  =    0.000;     /* Y distance from CS to CG */
double Cz1  =    0.000;     /* Z distance from CS to CG */
double m1   =    0.000;     /* mass */
double Ixx1 =    0.000;     /* Moments of Inertia */
double Iyy1 =  145.660;
double Izz1 =    0.000;
double Ixy1 =    0.000;     /* Products of Inertia */
double Ixz1 =    0.000;
double Iyz1 =    0.000;

/*---Initialize Aft Arm (Link 2) Parameters */

double Cx2  =  -36.000;     /* X distance from CS to CG */
```

```
double Cy2  =    0.000;     /* Y distance from CS to CG */
double Cz2  =    0.000;     /* Z distance from CS to CG */
double m2   =    0.776;     /* mass */
double Ixx2 =    5.830;     /* Moments of Inertia */
double Iyy2 = 1344.000;
double Izz2 = 1344.000;
double Ixy2 =    0.000;     /* Products of Inertia */
double Ixz2 =    0.000;
double Iyz2 =    0.000;

/*---Initialize Fore Arm (Link 3) Parameters */

double Cx3  =  -36.000;     /* X distance from CS to CG */
double Cy3  =    0.000;     /* Y distance from CS to CG */
double Cz3  =    0.000;     /* Z distance from CS to CG */
double m3   =    0.776;     /* mass */
double Ixx3 =    5.830;     /* Moments of Inertia */
double Iyy3 = 1344.000;
double Izz3 = 1344.000;
double Ixy3 =    0.000;     /* Products of Inertia */
double Ixz3 =    0.000;
double Iyz3 =    0.000;

/*---Initialize Nozzle (link 4) Parameters */

double Cx4  =  -23.250;     /* X distance from CS to CG */
double Cy4  =    0.000;     /* Y distance from CS to CG */
double Cz4  =    0.000;     /* Z distance from CS to CG */
double m4   =    0.388;     /* mass */
double Ixx4 =    2.620;     /* Moments of Inertia */
double Iyy4 =  339.530;
double Izz4 =  339.530;
double Ixy4 =    0.000;     /* Products of Inertia */
double Ixz4 =    0.000;
double Iyz4 =    0.000;

/*---Others */

double u[4];                /* Joint Torque */

/*---External Variables-------------------------------------------------*/
/*                                                                      */
/*  Cx.........X distance from CS to CG                                 */
/*  Cy.........Y distance from CS to CG                                 */
/*  Cz.........Z distance from CS to CG                                 */
/*  m..........Mass                                                     */
/*  Ixx........Moment of Inertia                                        */
/*  Iyy........                                                         */
/*  Izz........                                                         */
/*  Ixy........Product of Inertia                                       */
/*  Ixz........                                                         */
```

```
/*  Iyz.........                                                          */
/*  u...........Input Torques                                             */
/*                                                                        */
/*------------------------------------------------------------------------*/

main ()
{

/*---Passed Variables-----------------------------------------------------*/
/*                                                                        */
/*------------------------------------------------------------------------*/

  double h, t, tend, k1, k2, q[4], qd[4], y[4], yd[4];
  double pi = 3.141592654;
  int m;

/*---Local Variables------------------------------------------------------*/
/*                                                                        */
/*  m...........Number of equations to be integrated                      */
/*  h...........Integration Step Size                                     */
/*  t...........Simulation time                                           */
/*  tend........End Simulations Time                                      */
/*  k1..........Position Feedback gain                                    */
/*  k2..........Velocity Feedback gain                                    */
/*  q...........Joint Positions                                           */
/*  qd..........Joint Velocities                                          */
/*  pi..........Constant                                                  */
/*                                                                        */
/*------------------------------------------------------------------------*/

  extern double sin(), cos();

/*---External Variables---------------------------------------------------*/
/*                                                                        */
/*  sin.........Sine of an Angle                                          */
/*  cos.........Cosine of an Angle                                        */
/*                                                                        */
/*------------------------------------------------------------------------*/

  m = 8;                    /*  Number of equations to be integrated */

  h    = 0.05;              /*  Integration Step Size  */
  t    = 0.0;               /*  Initial Time  */
  tend = 5.0;               /*  End Simulation Time */
  k1   = 64.0;              /*  Position Error Feedback Gain */
  k2   = 16.0;              /*  Velocity Error Feedback Gain */

/*---Initial Position */

  y[0] =   72.0 * cos ( pi*t/10.0 ) + 12.0;
  y[1] =   72.0 * sin ( pi*t/10.0 ) + 12.0;
```

```
  y[2] = -24.75 + 48.0 * sin ( pi*t/10.0 ) + 12.0;
  y[3] =   pi/2.0 + 12.0*pi/180.0;

/*---Initial Global Velocity */

  yd[0] = -72.0*pi/10.0 * sin ( pi*t/10.0 );
  yd[1] =  72.0*pi/10.0 * cos ( pi*t/10.0 );
  yd[2] =  48.0*pi/10.0 * cos ( pi*t/10.0 );
  yd[3] =   0.0;

/*---Inverse Kinematics to get Initial Joint Positions and Velocities */

  inv_kin ( y, yd, q, qd );

  control ( m, h, t, tend, k1, k2, q, qd );

}
```

```
/* matrix:  Matrix and Vector Routines */


mult_matrix ( a, m, n, b, p, c )
double a[], b[], c[];
int m, n, p;
{
  register int i, j, k;
  double x,y;

  for ( i = 0; i < m; i++ )
    for ( j = 0; j < p; j++ )
    {
      x = 0.0;
      for ( k = 0; k < n; k++ )
      {
        x = x + *(a+n*i+k) * *(b+p*k+j);
      }
      c[p*i+j] = x;
    }
}



smult_matrix ( a, m, n, b, s )
double a[], b[], s;
int m, n;
{
  register int i;

  for ( i = 0; i < m*n; i++ )
    b[i] = *(a+i) * s;
}

add_matrix ( a, m, n, b, c )
double a[], b[], c[];
int m, n;
{
  register int i, j;

  for ( i = 0; i < m; i++ )
    for ( j = 0; j < n; j++ )
    {
      c[n*i+j] = *(a+n*i+j) + *(b+n*i+j);
    }
}

sub_matrix ( a, m, n, b, c )
double a[], b[], c[];
int m, n;
```

```
{
  register int i, j;

  for ( i = 0; i < m; i++ )
    for ( j = 0; j < n; j++ )
    {
      c[n*i+j] = *(a+n*i+j) - *(b+n*i+j);
    }
}

transpose_matrix ( a, m, n, at )
double a[], at[];
int m, n;
{
  register int i, j;

  for ( i = 0; i < m; i++ )
    for ( j = 0; j < n; j++ )
      at[m*j+i] = a[n*i+j];
}

print_matrix ( a, m, n )
int m, n;
double a[];
{
  register int i, j;

  for ( i = 0; i < m; i++ )
  {
    printf ("\n");
    for ( j = 0; j < n; j++ )
    printf ("%15.6f", *(a+n*i+j) );
  }
  printf ("\n");
}

zero_matrix ( a, m, n )
int m, n;
double a[];
{
  register int i;

  for ( i = 0; i < m*n; i++ )
    a[i] = 0.0;
}

idenity_matrix ( a, m, n )
int m, n;
double a[];
{
  register int i;
```

```
  zero_matrix ( a, m, n );
  for ( i = 0; i < m*n; i = i+n+1 )
    a[i] = 1.0;
}

cross ( a, b, c )
double a[3], b[3], c[3];
{
  c[0] =   a[1]*b[2] - b[1]*a[2];
  c[1] = - a[0]*b[2] + b[0]*a[2];
  c[2] =   a[0]*b[1] - b[0]*a[1];
}

double dot ( a, b )
double a[3], b[3];
{
  return ( a[0]*b[0] + a[1]*b[1] + a[2]*b[2] );
}

double norm ( a )
double a[3];
{
  extern double sqrt();
  return ( sqrt ( a[0]*a[0] + a[1]*a[1] + a[2]*a[2] ) );
}

unit_vector ( a, b )
double a[], b[];
{
  double x, norm ();

  x = norm ( a );
  b[0] = a[0] / x;
  b[1] = a[1] / x;
  b[2] = a[2] / x;
}

print_vector ( a )
double a[3];
{
  printf ("\n%15.6f\n%15.6f\n%15.6f\n", a[0], a[1], a[2] );
}
```

```
/* output:  Writes output data                                              */

/*--------------------------------------------------------------------------*/
/*                                                                          */
/*  Written By:  James A. Aardema                                           */
/*                                                                          */
/*  Date:  November 29, 1988                                                */
/*                                                                          */
/*  Modifications:                                                          */
/*                                                                          */
/*  Called by:  Control                                                     */
/*                                                                          */
/*  Language: C                                                             */
/*                                                                          */
/*  Compiler Options:  None                                                 */
/*                                                                          */
/*  Machine Dependencies:  None                                             */
/*                                                                          */
/*  Error: None                                                             */
/*                                                                          */
/*  Purpose:  Writes output data to the screen                              */
/*                                                                          */
/*--------------------------------------------------------------------------*/

/*---Header and Include Files---------------------------------------------*/
/*                                                                        */
/*---Symbolic Constants---------------------------------------------------*/
/*                                                                        */
/*------------------------------------------------------------------------*/

output ( t, y_d, y, yd_d, yd, q, qd, pos_error, vel_error, v, u )
double t, y_d[4], y[4], yd_d[4], yd[4], q[4], qd[4];
double pos_error[4], vel_error[4], v[4], u[4];
{

/*---Passed Variables-----------------------------------------------------*/
/*                                                                        */
/*  y_d.........Desired global position                                   */
/*  y...........Actual Position                                           */
/*  yd_d........Desired global velcity                                    */
/*  yd..........Actual Velocitiy                                         */
/*  q...........Joint Positions                                           */
/*  qd..........Joint Velocity                                            */
/*  pos_error...Position Error - ( Desired - Actual )                     */
/*  vel_error...Velocity Error - ( Desired - Actual )                     */
/*  v...........Control Input                                             */
/*  u...........Input Torques                                             */
/*                                                                        */
/*------------------------------------------------------------------------*/
```

```
/*---Local Variables-------------------------------------------------------*/
/*                                                                         */
/*-------------------------------------------------------------------------*/

/*---External Variables----------------------------------------------------*/
/*                                                                         */
/*-------------------------------------------------------------------------*/

  printf ("\n");
  printf ("Time    = %12.5f\n",  t );

  printf ("y_d     = %12.5f %12.5f %12.5f %12.5f\n",
                     y_d[0],       y_d[1],       y_d[2],       y_d[3] );

  printf ("y       = %12.5f %12.5f %12.5f %12.5f\n",
                     y[0],         y[1],         y[2],         y[3] );

  printf ("pos_err= %12.5f %12.5f %12.5f %12.5f\n",
                     pos_error[0], pos_error[1], pos_error[2], pos_error[3] );

  printf ("yd_d    = %12.5f %12.5f %12.5f %12.5f\n",
                     yd_d[0],      yd_d[1],      yd_d[2],      yd_d[3] );

  printf ("yd      = %12.5f %12.5f %12.5f %12.5f\n",
                     yd[0],        yd[1],        yd[2],        yd[3] );

  printf ("vel_err= %12.5f %12.5f %12.5f %12.5f\n",
                     vel_error[0], vel_error[1], vel_error[2], vel_error[3] );

  printf ("q       = %12.5f %12.5f %12.5f %12.5f\n",
                     q[0],         q[1],         q[2],         q[3] );

  printf ("qd      = %12.5f %12.5f %12.5f %12.5f\n",
                     qd[0],        qd[1],        qd[2],        qd[3] );

  printf ("v       = %12.5f %12.5f %12.5f %12.5f\n",
                     v[0],         v[1],         v[2],         v[3] );

  printf ("u       = %12.5f %12.5f %12.5f %12.5f\n",
                     u[0],         u[1],         u[2],         u[3] );
}
```

```
/* plot1:  Writes output data for plotting                                  */

/*--------------------------------------------------------------------------*/
/*                                                                          */
/*  Written By:  James A. Aardema                                           */
/*                                                                          */
/*  Date:  November 29, 1988                                                */
/*                                                                          */
/*  Modifications:                                                          */
/*                                                                          */
/*  Called by:  Control                                                     */
/*                                                                          */
/*  Language: C                                                             */
/*                                                                          */
/*  Compiler Options:  None                                                 */
/*                                                                          */
/*  Machine Dependencies:  None                                             */
/*                                                                          */
/*  Error: None                                                             */
/*                                                                          */
/*  Purpose:  Writes output data to the screen in columns for plotting      */
/*                                                                          */
/*--------------------------------------------------------------------------*/

/*---Header and Include Files---------------------------------------------*/
/*                                                                          */
/*---Symbolic Constants---------------------------------------------------*/
/*                                                                          */
/*--------------------------------------------------------------------------*/

plot1 ( t, y_d, y, yd_d, yd, q, qd, pos_error, vel_error, v, u )
double t, y_d[4], y[4], yd_d[4], yd[4], q[4], qd[4];
double pos_error[4], vel_error[4], v[4], u[4];
{

/*---Passed Variables-----------------------------------------------------*/
/*                                                                          */
/*  y_d.........Desired global position                                     */
/*  y...........Actual Position                                             */
/*  yd_d........Desired global velcity                                      */
/*  yd..........Actual Veloctiy                                            */
/*  q...........Joint Positions                                             */
/*  qd..........Joint Velocity                                              */
/*  pos_error...Position Error - ( Desired - Actual )                       */
/*  vel_error...Velocity Error - ( Desired - Actual )                       */
/*  v...........Control Input                                               */
/*  u...........Input Torques                                               */
/*                                                                          */
/*--------------------------------------------------------------------------*/
```

```
  double pi = 3.141592654;

/*---Local Variables-------------------------------------------------------*/
/*                                                                         */
/*  pi.........Constant                                                    */
/*                                                                         */
/*-------------------------------------------------------------------------*/

/*---External Variables----------------------------------------------------*/
/*                                                                         */
/*-------------------------------------------------------------------------*/

  printf ("%5.3f", t );

  printf (" %8.4f %8.4f %8.4f %8.4f",
           y_d[0],       y_d[1],       y_d[2],       y_d[3]*180.0/pi );

  printf (" %8.4f %8.4f %8.4f %8.4f",
           y[0],         y[1],         y[2],         y[3]*180.0/pi );

  printf (" %8.4f %8.4f %8.4f %8.4f",
           pos_error[0], pos_error[1], pos_error[2], pos_error[3]*180.0/pi );

  printf (" %8.4f %8.4f %8.4f %8.4f",
           vel_error[0], vel_error[1], vel_error[2], vel_error[3]*180.0/pi );

  printf ("\n");

}
```

```
/* plot2:  Writes output data for plotting                                  */

/*-------------------------------------------------------------------------*/
/*                                                                         */
/*  Written By:  James A. Aardema                                          */
/*                                                                         */
/*  Date:  November 29, 1988                                               */
/*                                                                         */
/*  Modifications:                                                         */
/*                                                                         */
/*  Called by:  Control                                                    */
/*                                                                         */
/*  Language: C                                                            */
/*                                                                         */
/*  Compiler Options:  None                                                */
/*                                                                         */
/*  Machine Dependencies:  None                                            */
/*                                                                         */
/*  Error: None                                                            */
/*                                                                         */
/*  Purpose:  Writes output data to the screen in columns for plotting     */
/*                                                                         */
/*-------------------------------------------------------------------------*/

/*---Header and Include Files--------------------------------------------*/
/*                                                                       */
/*---Symbolic Constants-------------------------------------------------*/
/*                                                                       */
/*-----------------------------------------------------------------------*/

plot2 ( t, y_d, y, yd_d, yd, q, qd, pos_error, vel_error, v, u )
double t, y_d[4], y[4], yd_d[4], yd[4], q[4], qd[4];
double pos_error[4], vel_error[4], v[4], u[4];
{

/*---Passed Variables---------------------------------------------------*/
/*                                                                      */
/*  y_d.........Desired global position                                 */
/*  y...........Actual Position                                         */
/*  yd_d........Desired global velcity                                  */
/*  yd..........Actual Velocti                                          */
/*  q...........Joint Positions                                         */
/*  qd..........Joint Velocity                                          */
/*  pos_error...Position Error - ( Desired - Actual )                   */
/*  vel_error...Velocity Error - ( Desired - Actual )                   */
/*  v...........Control Input                                           */
/*  u...........Input Torques                                           */
/*                                                                      */
/*----------------------------------------------------------------------*/
```

```
  double pi = 3.141592654;

/*---Local Variables-------------------------------------------------------*/
/*                                                                         */
/*  pi.........Constant                                                    */
/*                                                                         */
/*-------------------------------------------------------------------------*/

/*---External Variables----------------------------------------------------*/
/*                                                                         */
/*-------------------------------------------------------------------------*/

  printf ("%5.3f", t );

  printf (" %6.4f %6.4f %6.4f %6.4f",
            q[0],          q[1],          q[2],          q[3] );

  printf (" %8.4f %8.4f %8.4f %8.4f",
            qd[0],         qd[1],         qd[2],         qd[3] );

  printf (" %8.3f %8.3f %8.3f %8.3f",
            v[0],          v[1],          v[2],          v[3] );

  printf (" %8.3f %8.3f %8.3f %8.3f",
            u[0],          u[1],          u[2],          u[3] );

  printf ("\n");

}
```

```
/* rk4_step:  Integrate a system of first order differential equations     */
/*            over 1 time step using 4th order Runga-Kutta                */

/*--------------------------------------------------------------------------*/
/*                                                                          */
/*  Written By:  James A. Aardema                                           */
/*                                                                          */
/*  Date:  October 10, 1988                                                 */
/*                                                                          */
/*  Modifications:                                                          */
/*                                                                          */
/*  Called by:                                                              */
/*                                                                          */
/*  Language: C                                                             */
/*                                                                          */
/*  Compiler Options:  None                                                 */
/*                                                                          */
/*  Machine Dependencies:                                                   */
/*                                                                          */
/*  Error:                                                                  */
/*                                                                          */
/*  Purpose:  This subroutine integrates a system of "m" first order        */
/*  initial value problems using a 4th order Runga-Kutta Algorithm.         */
/*                                                                          */
/*  The system of equations is integrated form (t) to (t+h).  The values    */
/*  t is updated upon return.  The vector "y" contains the initial          */
/*  conditions upon entry and contains the final values upon output.        */
/*  The routine is designed to be called repeatedly, in a for or while      */
/*  loop with with minimum effort.                                          */
/*                                                                          */
/*  A working vector "w" of length ( 5 * m ) is required.                   */
/*                                                                          */
/*  For a complete description on the theory and development of this        */
/*  algorithm see page 264 of the book "Numerical Analysis" by              */
/*  Richard L. Burden and J. Douglas Faires.                                */
/*                                                                          */
/*--------------------------------------------------------------------------*/

/*---Header and Include Files----------------------------------------------*/
/*                                                                          */
/*---Symbolic Constants----------------------------------------------------*/
/*                                                                          */
/*--------------------------------------------------------------------------*/

rk4_step ( m, y, w, h, t )
double y[], w[], *h, *t;
int m;
{

/*---Passed Variables------------------------------------------------------*/
```

```
/*                                                                          */
/*  m..........Number of equations to be Integrated                         */
/*  y..........Initial Conditions on input; Final conditions on Output      */
/*  w..........Working vector of length "5*m"                               */
/*  h..........Integration Step Size                                        */
/*  t..........Current Integration Time                                     */
/*                                                                          */
/*--------------------------------------------------------------------------*/

  double x, *k1, *k2, *k3, *k4;
  register int j;

/*---Local Variables-------------------------------------------------------*/
/*                                                                          */
/*  x..........Local Value for Time                                         */
/*  k1.........Working Vector                                               */
/*  k2.........Working Vector                                               */
/*  k3.........Working Vector                                               */
/*  k4.........Working Vector                                               */
/*  j..........Index                                                        */
/*                                                                          */
/*--------------------------------------------------------------------------*/

/*---External Variables----------------------------------------------------*/
/*                                                                          */
/*--------------------------------------------------------------------------*/

  k1 =  w + m;          /*  Set working vector of lenght "m" */
  k2 = k1 + m;          /*  Set working vector of length "m" */
  k3 = k2 + m;          /*  Set working vector of lenght "m" */
  k4 = k3 + m;          /*  Set working vector of lenght "m" */

  /* Step 5 */

  diffeq ( k1, *t, y, m );

  /* Step 6 */

  x = *t + *h/2.0;
  for ( j = 0; j < m; j++ ) w[j] = y[j] + *h*k1[j]/2.0;
  diffeq ( k2, x, w, m );

  /* Step 7 */

  for ( j = 0; j < m; j++ ) w[j] = y[j] + *h*k2[j]/2.0;
  diffeq ( k3, x, w, m );

  /* Step 8 */

  x = *t + *h;
  for ( j = 0; j < m; j++ ) w[j] = y[j] + *h*k3[j];
```

```
    diffeq ( k4, x, w, m );

    /* Step 9 */

    for ( j = 0; j < m; j++ )
      y[j] = y[j] + *h*( k1[j] + k2[j] + k2[j] + k3[j] + k3[j] + k4[j]) / 6.0;

    /* Step 10 */

    *t = *t + *h;

}
```

```
/* trajectory:  Defines the robot trajectory                                */

/*--------------------------------------------------------------------------*/
/*                                                                          */
/*  Written By:  James A. Aardema                                           */
/*                                                                          */
/*  Date:  November 29, 1988                                                */
/*                                                                          */
/*  Modifications:                                                          */
/*                                                                          */
/*  Called by:                                                              */
/*                                                                          */
/*  Language: C                                                             */
/*                                                                          */
/*  Compiler Options:  None                                                 */
/*                                                                          */
/*  Machine Dependencies:  None                                             */
/*                                                                          */
/*  Error: None                                                             */
/*                                                                          */
/*  Purpose:  Define the trajectory of the nozzle ( Coordinate System 4 )   */
/*                                                                          */
/*--------------------------------------------------------------------------*/

/*---Header and Include Files---------------------------------------------*/
/*                                                                          */
/*---Symbolic Constants---------------------------------------------------*/
/*                                                                          */
/*--------------------------------------------------------------------------*/

trajectory ( t, y_d, yd_d, ydd_d )
double t, y_d[4], yd_d[4], ydd_d[4];
{


/*---Passed Variables-----------------------------------------------------*/
/*                                                                          */
/*  t..........Time                                                         */
/*  y_d........Desired Position                                             */
/*  yd_d.......Desired Velocity                                             */
/*  ydd_d......Desired Accelerations                                        */
/*                                                                          */
/*--------------------------------------------------------------------------*/

  double x;
  double pi = 3.141592654;

/*---Local Variables------------------------------------------------------*/
/*                                                                          */
/*  x..........Scaled Value of Time                                         */
```

```
/*  pi.........Constant                                                    */
/*                                                                         */
/*-------------------------------------------------------------------------*/

  extern double sin(), cos();

/*---External Variables--------------------------------------------------*/
/*                                                                         */
/*   sin........Sine of an angle                                           */
/*   cos........Cosine of an angle                                         */
/*                                                                         */
/*-------------------------------------------------------------------------*/

/*---Desired Trajectory Position */

  y_d[0] =   72.0 * cos ( pi*t/10.0 );
  y_d[1] =   72.0 * sin ( pi*t/10.0 );
  y_d[2] = -24.75 + 48.0 * sin ( pi*t/10.0 );
  y_d[3] =   pi/2.0;

/*---Desired Trajectory Velocity */

  yd_d[0] = -72.0*pi/10.0 * sin ( pi*t/10.0 );
  yd_d[1] =  72.0*pi/10.0 * cos ( pi*t/10.0 );
  yd_d[2] =  48.0*pi/10.0 * cos ( pi*t/10.0 );
  yd_d[3] =   0.0;

/*---Desired Trajectory Acceleration */

  ydd_d[0] = -72.0*pi*pi/100.0 * cos ( pi*t/10.0 );
  ydd_d[1] = -72.0*pi*pi/100.0 * sin ( pi*t/10.0 );
  ydd_d[2] = -48.0*pi*pi/100.0 * sin ( pi*t/10.0 );
  ydd_d[3] =   0.0;

}
```

DISTRIBUTION LIST

	Copies
Commander U.S. Army Tank-Automotive Command ATTN: ASNC-TAC-DIT (Technical Library) AMSTA-CF (Dir. RDE center) AMSTA-CR (Technical Director) AMSTA-NL (Logistics Technology Office) AMSTA-R (Dir. Tank-Automotive Technology) AMSTA-RR (Robotics Division) AMSTA-RY (System Simulation and Tech. Div.) AMSTA-U (Dir. for Systems Eng.) AMSTA-Z (Concepts & Tech. Demonstration Div.) Warren, MI 48397-5000	 2 1 1 1 1 1 20 1 3
Commander Defense Technical Information Center Bldg. 5, Cameron Station ATTN: DDAC Alexandria, VA 22304-9990	12
Manager Defense Logistics Studies Information Exchange ATTN: AMXMC-D Fort Lee, VA 23801-6044	2
Commander U.S. Army Operational Test and Evaluation Agency ATTN: CSTE-CS (Combat Support Div.) 5600 Columbia Pike Falls Church, VA 22041-5115	1
Deputy Commander U.S. Army Strategic Defense Command P.O. Box 1500 ATTN: DASD-H-S (Systems Anal./Battle Management Dir.) DASD-H-V (Advanced Technology Dir.) Huntsville, AL 35807-3801	2
Commander U.S. Army Foreign Science & Tech Center 220 Seventh Street NE ATTN: AIAST-RA (Research and Analysis Dir.) Charlottesville, VA 22901-5396	1

DISTRIBUTION LIST (Continued)

	Copies
Director U.S. Army Cold Regions Research & Engineering Lab P.O. Box 282 ATTN: CECRL-IC (Technical Library) Hanover, NH 03755-1290	1
Director U.S. Army Corps of Engineers Waterways Experiment Station P.O. Box 631 ATTN: WESGV-Z (Geotechnical Laboratory) Vicksburg, MS 39180-0631	1
U.S. Army Laboratory Command Army Research Office P.O. Box 12211 ATTN: SLCRO-EG (Engineering Division) SLCRO-TS (Library Services) Research Triangle Park, NC 27709-2211	2
Director U.S. Army Ballistic Research Lab ATTN: SLCBR-D (Office of Director) Aberdeen Proving Grounds, MD 21005-5066	1
Director Harry Diamond Laboratories ATTN: SLCHD-IT (Eng. & Tech Support Div.) SLCHD-TA (Tech. Applications Div.) 2800 Powder Mill Road Adelphi, MD 20783-1197	2
Director U.S. Army Human Engineering Laboratory ATTN: SLCHE-D (Office of Director) SLCHE-SS-TS (Library) Aberdeen Proving Grounds, MD 21005-5001	2
Commander U.S. Army Material Command ATTN: AMCDE (Development, Eng, & Acqusition) AMCDMA-ML (Library) 5001 Eisenhower Avenue Alexandria, VA 22333-001	2

DISTRIBUTION LIST (Continued)

	Copies
Commander U.S. Army Armament Munitions and Chemical Command ATTN: SMCAR-MS (Information Management Dir. - Tech Library) Picatinny Arsenal, NJ 07806-5000	1
Commander St. Louis Support Command ATTN: DACL/ASMC-STL-DACL (Library/Information Center) Granite City, IL 62040-1801	1
Director CECOM Research Development & Engineering Center ATTN: AMSEL-RD (Director) Fort Monmouth, NJ 07703-5001	1
Commander U.S. Army Missile Command ATTN: AMCPM (Land Combat Systems) AMSMI-CG (Reference Library) AMSMI-RD (Research, Dev. and Eng. Center) Redstone Arsenal, AL 35898-5000	3
Commander U.S. Army Cold Regions Test Center ATTN: STECR-MT (Materiel Test Dir.) Fort Greenly, AL	1
Commander U.S. Army Combat Systems Test Activity ATTN: Armament Systems Directorate Close Combat Systems Directorate Engineering Directorate Development & Analysis Directorate Aberdeen Proving Grounds, MA 21005-5059	4
Commander U.S. Army Yuma Proving Ground ATTN: STEYP-MT (Material Test Dir.) Yuma, AX 85364	1
Director TRAC-FLVN ATTN: ATRC-F Fort Leavenworth, KS 66027-5200	1

DISTRIBUTION LIST (Continued)

	Copies
Director TRAC-WSMR ATTN: ATRC-W White Sands Missile Range, NM 88002-5502	1
Commander U.S. Army Center and Fort Knox ATTN: ATZK-DOTD (USAARMS Library) ATZK-CD (Dir. of Combat Developments) ATZK-AE-AO (Library) Fort Knox, KY 40121-5000	3
President U.S. Army Armor and Engineer Board Fort Knox, KY 40121-5470	1
President US Army Aviation Board ATTN: ATZQ-OTC (Technical/Operations Div.) Fort Rucker, AL 36362-5064	1
Director U.S. Army Combat Developments Experimentation Center ATTN: Technical Library; Bldg 2925 Fort Ord, CA 93941-7000	1
Commander U.S. Army Combined Arms Center and Fort Leavenworth ATTN: ATOR-CT (Scientific and Tech. Support Dir.) ATZL-CA (Combined Arms Combat Devl. Activity) Fort Leavenworth, KS 66027-5130	2
Commander TRADOC Combined Arms Test Activity ATTN: ATCT-MA (Methodology & Analysis Dir.) ATCT-SPT (Technical Library) Fort Hood, TX 76544-5065	2
President U.S. Army Communications-Electronics Board ATTN: ATZH-BDS (Analysis & Tech. Support Div.) Fort Gordon, GA 30905-5350	1
Commander U.S. Army Engineer Center and Fort Belvoir ATTN: ATZA-SS-LIB (Library) Fort Belvoir, VA 22060-5000	1

DISTRIBUTION LIST (Continued)

	Copies
President U.S. Army Field Artillery Board ATTN: ATZR-BDS (Analysis and Tech. Support Div.) Fort Sill, OK 73503-6100	1
President U.S. Army Infantry Board ATTN: ATZB-IB-PR-T (Tech/Science Dir.) Fort Benning, GA 31905-5800	1
Commander U.S. Army Infantry School ATTN: ATSH-CD (Dir. of Combat Development) Fort Benning, GA 31905-5000	1
Commander U.S. Army Intelligence Center and School ATTN: ATSI-CD (Dir. of Combat Developments) Fort Huachuca, AZ 85613-7000	1
President/Commander U.S. Army Intelligence and Security Board ATTN: ATSI-BD-A (Analysis & Tech. Support Div.) Fort Huachuca, AZ 85613-7000	1
Commander U.S. Army Ordance Center and School ATTN: ATSL-CD (Dir. of Combat Development) ATSL-SE-LI (Library) Aberdeen Proving Ground, MD 21005-5201	2
Commander U.S. Army Ordance Missile and Munitions Center and School ATTN: ATSK-C (Dir. of Combat Developments) ATSK-AB (Library) Redstone Arsenal, AL 35897-6500	2
Headquarters U.S. Army Quartermaster Center and Fort Lee ATTN: ATSM-CD (Dir. of Combat Developments) Fort Lee, VA 23801-500095,96	1

DISTRIBUTION LIST (Continued)

	Copies
Headquarters U.S. Army Troop Support Command ATTN: AMSTR-B (Systems Analysis Office) AMSTR-E (Research & Development Integration Office) 4300 Goodfellow Blvd. St. Louis, MO 63120-1798	2
Commander US Army Belvoir Research, Development, & Engineering Center ATTN: STRBE-BT (Technical Library Div) Fort Belvoir, VA 22060-5606	1
Commander U.S. Army Natic Research, Development, and Engineering Center ATTN: STRNC-ML (Technical Library) Natick, MA 01760-5000	1
Commander U.S. Army Development and Readiness Command 5001 Eisenhower Avenue ATTN: Dr. R. S. Wiseman Alexandria, VA 22333	1
Commander US Army Soldier Support Center National Capitol Region ATTN: ATNC-NMM-A 200 Stovall Street Alexandria, VA 22332-0400	1
HQDA Office of Dep Chief of Staff for Rsch Dev & Acquisition ATTN: ARZ-A Dr. Lasser - Dir. of Army Research DAMA-AR Washington, D.C. 20310	2
Commander U.S. Army Military Equipment R&D Command ATTN: DRDME-T Fort Belvoir, VA 22060	1
Commandant U.S. Army Engineer School ATTN: ATZA-CDT Fort Belvoir, VA 22060-5281	1

DISTRIBUTION LIST (Continued)

	Copies
Commander Rock Island Arsenal ATTN: SARRI-LR Rock Island, IL 61201	1
Director U.S. Army Material Systems Analysis Agency ATTN: AMXSY-DD AMXSY-C (Mr. Harold Burke) AMXSY-CM (Mr. Fordyce) AMXSY-MP (Mr. Cohen) Aberdeen Proving Ground, MD 21005-5071	4
Director Keweenaw Research Center Michigan Technological University Houghton, MI 49931	1
Director Defense Advanced Research Projects Agency 1400 Wilson Boulevard Arlington, VA 22209	1
Commander U.S. Army Materials and Mechanics Research Center ATTN: Mr. Adachi Watertown, MA 02172	1
Director U.S.D.A. Forest Service Equipment Development Center 444 East Bonita Avenue San Dimes, CA 91773	1
Engineering Society Library 345 East 47th Street New York, NY 10017	1
Commander U.S. Army Armament Research and Development Command ATTN: Mr. Rubin Dover, NJ 07801	1
Commander USAMC Material Readiness Support Activity ATTN: AMXMD-ED (Equip Develop/Deploy Anal Br) Lexington, KY 40511-5101	1

DISTRIBUTION LIST (Continued)

	Copies
Commander U.S. Army Logistics Center ATTN: ATCL-M Fort Lee, VA 23801-6000	1
Director TRADOC Systems Analysis Activity ATTN: ATOR-TF White Sands Missile Range, NM 88002-5502	1
Commander U.S. Army Test Evaluation Command ATTN: AMSTE-BB AMSTE-TA AMSTE-TE AMSTE-TD-T Aberdeen Proving Ground, MD 21005-5055	4
Commanding General U.S. Marine Corps Mobility & Logistics Division Development and Education Command ATTN: Mr. Hickson MCOTEA Quantico, VA 22134	2
Commander U.S. Air Force Tactical Air Warfare Center ATTN: TE Eglin AFB, FL 32542	1
Commandant U.S. Army Armor School ATTN: ATSB PERI-IK Fort Knox, KY 40121-5000	2
Commander US Army Transportation School ATTN: ATSP-TD Fort Eustis, VA 23604-5361	1
Commander US Army Training Support Center ATTN: ATIC-DM Fort Eustis, VA 23604-5166	1

DISTRIBUTION LIST

	Copies
President CDED Test Board ATTN: ATEC-B Fort Lewis, WA 98433-5000	1
Superintendent U.S. Military Academy ATTN: Dept. of Engineering West Point, NY 10996	1